MULTISTRATEGY CONSTRUCTIVE INDUCTION

by

Eric E. Bloedorn

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
the Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

______________________	Ryszard S. Michalski, Dissertation Director
______________________	Kenneth De Jong
______________________	Larry Kerschberg
______________________	Gheorghe Tecuci
______________________	Carl Harris, Interim Associate Dean for Graduate Studies and Research
______________________	W. Murray Black, Interim Dean, School of Information Technology and Engineering

Date: ______________

Fall, 1996
George Mason University
Fairfax, Virginia

Multistrategy Constructive Induction

A dissertation submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy at George Mason University

By

Eric E. Bloedorn
B.A., Lawrence University, 1989
M.S., George Mason University, 1992

Director: Dr. Ryszard S. Michalski, Professor
Department of Computer Science and Systems Engineering

Fall 1996
George Mason University
Fairfax, Virginia

CHAPTER 1 INTRODUCTION

Machine learning (ML) algorithms are increasingly being pressed into service to help users understand and detect patterns or regularities found in large amounts of data. These tools are needed to help human analysts make sense of the increasing amount of complex data available electronically from domains as diverse as computer vision, to world economics. One of the primary difficulties preventing these ML algorithms from accurately and simply describing the data is that all detected patterns must be stated in terms of attributes and values explicitly found in the representation. When the data is complex and contains irrelevant attributes, too much precision in the values, or inter-dependent attributes, current approaches fail to find simple, accurate patterns. One method for overcoming the problem of a poor representation is to simultaneously search for patterns, and for an improved representation of the problem itself. This line of research is known as constructive induction (CI). Extending previous work on CI which was primarily concerned with expanding the representation space by adding new terms, this thesis introduces an implementation and framework for multistrategy constructive induction (MCI). The MCI approach contains methods to not only expand the representation by adding new problem-relevant attributes, but also to contract the representation space by removing attributes and/or attribute-values.

1.1 Background

Learning is a central element of intelligence. With learning previous obstacles can be overcome, previous approaches can be improved, and new problems may be solved quickly. Without learning past mistakes will be repeated and success will require either careful constraints on the problem, or constant supervision. Because of its central role in intelligent behavior, learning has been widely studied and is of interest to such varied fields as philosophy, cognitive science, psychology, education, information science and artificial intelligence.

The emphasis in this thesis is on Machine Learning (ML). Machine learning is the study of how learning processes can be automated, or modelled on a computer. One motivation for doing this research is to build artificial systems that can autonomously perform some task or to act as intelligent assistants to human users in performing complex tasks. Application areas of machine learning are diverse and include such topics as computer vision, economics, medicine, and engineering.

Although there are numerous applications and many domain specific techniques, much research in ML is focused on general learning techniques. The work described here is more general and can be categorized into the broad branch of techniques for "learning from examples." Learning from examples is a problem of intermediate difficulty when all possible learning problems are ranked based on the amount of effort which is required of the learner (Carbonell, et al., 1983). This ranking is shown below:

1. Rote learning, memorization or the direct implantation of knowledge
2. Learning from instruction or learning by being told
3. Learning from analogy
4. Learning from examples
5. Learning from observation and discovery

In "Rote learning" the learner has to perform little manipulation of the information given before it is useful in solving a problem. The difficulty for the learner increases as one moves to "Learning from instruction", "Learning from analogy", "Learning from examples", and finally "Learning from observation and discovery". As one moves from one to five, the learner has an increasing amount of responsibility and autonomy. The focus of this dissertation is on the class of machine learning algorithms which perform *learning from examples,* or supervised classification. In this type of learning, an external source or expert has assigned class labels to a set of examples. The learner's goal is to generate a hypothesis which discriminates each class from all others. Although it could be performed when the class labels of all possible examples are known, the most common case of learning from examples occurs when a learner is presented with an incomplete set of known cases and is asked to generate general hypotheses from them. These generalizations induced from the given examples describe the conditions under which one class of examples is different from other classes (in the case of discriminant descriptions) or different from *any* other class (in the case of characteristic descriptions). Such generalizations can be used to predict the class of unseen examples. However, because the generalizations are obtained through an inductive transformation, they may not correctly predict all of the new cases. One factor that plays an important role in determining the predictive accuracy of learned hypotheses is the representation of the problem.

1.2 Importance of the Problem

In learning from examples the first step is to build a representation of the problem. The representation space of a problem is the space of possibilities defined by the domains of the given descriptors. It is important that the representation space captures the necessary information to characterize or discriminate the given classes of examples. In some cases the important descriptors are known, but in many cases not. Take the problem of predicting when a stock will increase in value. Is the previous day's price important? What about the price or change in price of related stocks? What about other markets? Once the descriptors are selected, then the granularity of the problem must be determined. Should values be stated in terms of tens, hundreds or thousands of dollars? If the granularity is too fine, then general trends in the data may be lost in local fluctuations. Conversely a coarse granularity can result in small patterns being missed.

Once a representation has been determined, the goal of supervised classification methods is to select those factors or conditions that are important in discriminating (or characterizing) between the different classes of examples (e.g. differentiating between good times to sell stocks from bad). Empirical methods for generating hypotheses look for the similarities and differences between the like-labeled examples provided. The resulting hypotheses are generalizations of these findings. If few examples exist, or the features are only weakly relevant to the problem, the resulting patterns found may only coincidentally cover the data and not represent patterns true for the entire set. For this reason the representation of the problem has a profound effect on the performance of the learning algorithm.

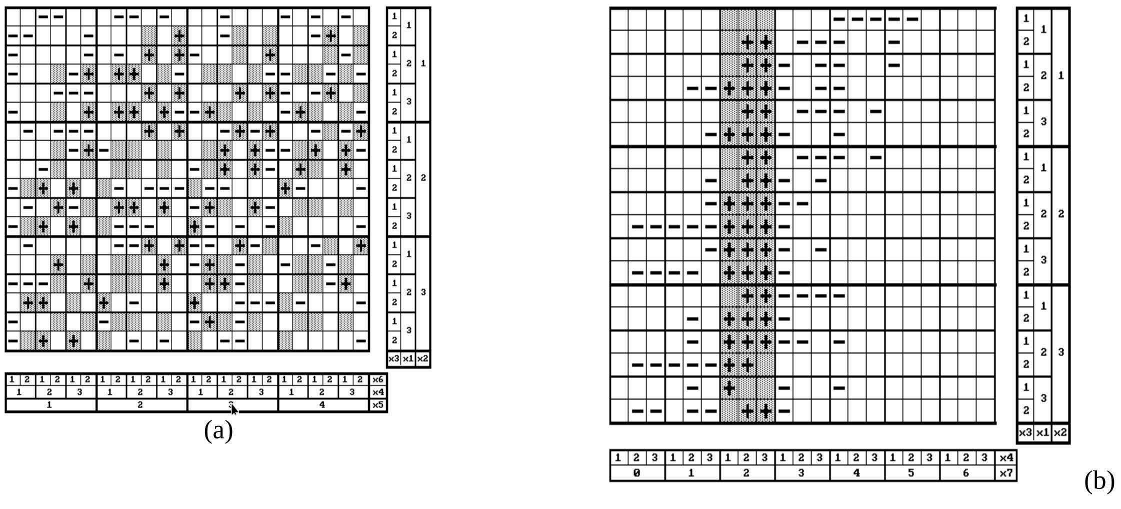


Figure 1.1 - Diagrammatic visualization of the Monk2 representation space before (a) and after (b) data-driven constructive induction

The effect of representation on the quality of the generated hypotheses is well illustrated by the second Monk’s problem (Thrun, 1991). This problem is difficult because it has a distributed coding, similar to parity problems. The goal concept is given as: An example is in class 1 if exactly two of the six attributes have their first value. The original representation space with the training examples in + and -, and the target concept shaded is shown in Figure 1.1(a) using DIAV (Wnek, 1995).

The learning problem as represented in Fig. 1.1(a) is clearly difficult. There appears to be no simple pattern or description. In fact, the rule shown in the shaded region is only 77% accurate in predicting the class of unseen examples. On the other hand the problem of Figure 1.1(b) is much easier to cover and describe. In this case the '+' squares (positive examples) can be covered by a simple rule: [x7=2] => square is a positive example. The improved representation space of Figure 1.1(b) was found by the program AQ17-DCI (Bloedorn, 1996). The transformation made by AQ17-DCI was to generate a new attribute using the a counting function called #VarEQ(x) function. This operator builds a new attribute for all values of x found in the data. In this problem the most useful new attribute was #VarEQ(1) which counts the number of attributes that take their first value. the value. The transformed representation space including this attribute (x7 in the figure) is shown in Figure 1.1(b). (Attributes x5 and x6 were removed for clarity). In this new representation space generated rules were simple and had a predictive accuracy of 100%.

1.3 Constructive Induction

In recognition of the important step of finding the appropriate representation space for a problem, the idea of constructive induction (CI) was introduced (Michalski, 1983). This initial definition of CI focused solely on the capability for "formulating new descriptors". Many programs which perform CI today including LFC (Ragavan and Rendell, 1993), CITRE

(Matheus, 1989), FRINGE (Pagallo and Haussler, 1990) and GALA (Hu and Kibler, 1996), still view CI in this way. However, the definition of CI has recently been extended by some to include *any* change in the representation space (Wnek and Michalski, 1994; Bloedorn and Michalski, 1996). In this new view, CI includes both expanders which enlarge the space by formulating new descriptors, and contractors which reduce the space by removing descriptors, or descriptor values. In this new view the process of learning is seen as two intertwined searches. The first search is for a ‘good’ representation space. This step was previously performed by the domain experts by themselves or in collaboration with experts in machine learning which articulated the problem. The second search is for a hypothesis which discriminates between the classes of examples given in the problem. This search can be performed by any one of a number of learning methods. Well known programs for learning hypotheses from examples include C4.5 (Quinlan, 1993), Backpropagation (Rummelhart and McClelland, 1986), CN2 (Clark and Niblett, 1989) and AQ15c (Wnek, 1995), the program used in the method described here.

The basic premise of research on constructive induction (CI) is that a precondition for satisfactory learning results is a well-stated representation of the problem. If the representation space is well chosen, then the results of learning will be satisfactory with almost any learning method. Conversely, with a poor representation learning will be poor regardless of the method. Constructive induction is oriented toward learning problems in which the representation space defined by the training examples is of low quality. This occurs when the space contains weakly relevant or irrelevant attributes, or there is a mismatch between the description language and the target concept in the originally given representation space. Stated another way, Constructive Induction can also be thought of as automatically determining an adequate “representation space bias” for learning.

The search for an improved representation space can be guided by information from three sources (Wnek and Michalski, 1994) training data (as in data-driven constructive induction—DCI), initial hypotheses learned from the data (as in hypothesis-driven constructive induction—HCI), or expert knowledge provided by the user to the system (as in knowledge-driven constructive induction—KCI). These approaches can also be combined into a multistrategy constructive induction method (MCI). This thesis describes such a multistrategy constructive induction method.

1.4 Potential Benefits

There are many potential benefits of a system which can improve the current representation space without explicit expert guidance. The benefits of improved representation spaces include:

1) Increased predictive accuracy of learned hypotheses. When the representation space is improved the learner can detect the patterns that better describe the given examples and can thus better predict new examples.

2) Decreased complexity. Although accurate hypotheses can be learned from a poor representation such learned hypotheses are often unnecessarily complex. Comprehensibility is often important, especially when learning hypotheses for later use by people. Complexity is measured in the experiments in Chapter 5 by the number of selectors and the number of rules in the learned hypotheses. Chapter 5 also shows examples of significant decreases in complexity due to changes made by MCI.

3) Decreased learning time. Decreased learning time is possible when the examples are arranged in a pattern which is easy to describe for the chosen hypothesis language. Modifications to the representation space, as shown in Figure 1.1, and in Chapter 5, can significantly simplify the search for good hypotheses, and thus reduce learning time. Learning time is measured in the experiments in seconds.

4) Higher Noise tolerance. When the representation space is well suited to learning, high levels of misclassification noise can be tolerated without decrease in predictive accuracy. This is due to the way the changes to the representation have better grouped like-labelled examples into easier to describe clusters. Although valuable in some domains, this effect is not directly tested in this thesis.

5) Fewer learning examples required. Fewer learning examples are required in improved representation spaces because the remaining examples still clearly define the best hypotheses. This effect is also not emprically validated here, but is clear from cases like Monk 2 shown previously.

6) Improved performance on related problems. One of the control strategies introduced here for multistrategy constructive induction method stores a record of its performance on past examples which helps it perform better on related future problems. This memory allows the system to learn how to improve the representation space of future problems so that future learned rules are of high quality.

7) Improved performance on single problems with multiple pathologies. Most real-world induction problems are difficult because they have a mixture of inappropriately represented attributes, or over-precise attribute values or irrelevant attributes. As shown in chapter 5, multistrategy constructive induction can improve on even these multi-pathological cases.

Some of these benefits have been empirically shown by other researchers, but usually for a narrow range of synthetic problems. Chapter 5 will demonstrate most of these benefits on both synthetic and two real-world problems.

1.5 Thesis

Multistrategy Constructive Induction (MCI) is a solution to the problem of learning simple, predictively accurate rules from examples in representations which contain irrelevant attributes, inter-dependent attributes, and large attribute domains. MCI is the only method to combine representation space modification operators (RSMOs) to address all of these difficulties. This allows MCI to solve problems that no single-strategy method alone can solve. The MCI framework presented here can include representation space modification operators which make use any possible computational strategy. The implementation presented here makes use of heuristics, statistics and learned hypotheses. Methods which use other knowledge-driven and evolutionary computation techniques are also outlined. A method for meta-level learning in which the system itself acquires the control rules necessary for determining which RSMO is best suited for the given problem is also outlined and shown to be a useful and effective control strategy.

The defense of this thesis is organized as follows: Chapter 2 provides background and motivation for this research. This chapter provides definitions of important terms and describes related work. Chapter 3 defines multistrategy constructive induction, the tools combined, and their various control strategies. An analysis of the links between MCI and the Inferential Theory of Learning (ITL) (Michalski, 1993) is given to provide a deeper understanding of the transformations being performed. The architecture of multistrategy constructive induction as implemented in AQ17-MCI is given in Chapter 4. This is followed by an analysis of the relationship between MCI and the Inferential Theory of Learning (Michalski, 1993). Chapter 5 describes the set of experiments designed to evaluate the effectiveness of the proposed approach. The last chapter, Chapter 6, provides a summary of results and an identification and analysis of outstanding research problems.

1.6 Major Contributions

This dissertation introduces a novel methodology for multistrategy constructive induction. The major contributions of this approach are 1) it incorporates multiple computational methods for constructive induction including representation space expansion and contraction, 2) it incorporates multiple inferential techniques. It uses deduction to arrive at a meta-decision concerning which representation space modifier to select, and induction when inducing a new (or modifying a previous) meta-rule from a set of meta-examples, and 3) it is a learning system capable of improving its own performance over time through meta-learning. The proposed method is built on established individual empirical induction and constructive induction techniques and is capable of incorporating knowledge from many sources including 1) directly from the user, 2) from analysis of the data, and 3) from analysis of learned hypotheses.

The proposed multistrategy approach helps to overcome the brittleness of current learning methods by automating the search for representation spaces which are better suited to learning predictively accurate rules. This approach helps overcome the problems such methods have with complex real-valued data by including a method for attribute-value discretization, and with noisy data, by including many methods for representation space contraction. The meta-learning capabilities eliminate the need for the need for human expertise to guide the selection of these tools. The relationship between characteristics of a dataset and appropriate representation space transformations are not generally known. A learning approach to this meta-learning task eliminates the need to explicitly determine this relationship before using the available tools.

CHAPTER 2 BACKGROUND AND MOTIVATION

This chapter provides background and motivation for research in multistrategy constructive induction. Definitions of learning from examples, representation space and constructive induction are given. This is followed by an analysis of why current selective inductive learning algorithms fail and how these failures are currently addressed.

2.1 Definitions

2.1.1 Learning From Examples

Multistrategy constructive induction performs both a search for an adequate representation space with which to represent examples, and a search for hypotheses which describe the examples. The latter search is performed as an inductive inference and is a form of learning from examples. The problem of learning from examples is defined as follows (Michalski, 1983):

Given:

* A set of observational statements (facts), F, that represent a collection of implications where each statement denotes a description of an example of concept or class K_i and i is a set indexing classes K_i
* A tenative inductive assertion (which maybe null),
* Background knowledge that defines the assumptions and constraints imposed on the observational statements and generated candidate inductive assertions, and any relevant problem domain knowledge. The last includes the preference criterion characterizing the desirable properties of the sought inductive assertion

Find:

* A set of concept recognition rules, H, H: $\{D_i ::> K_i\}$, i I, that are consistent with the background knowledge

where D_i is a concept description of class K_i, or a description of the conditions under which, if satisfied, an object is considered an instance of class K_i. A simple learning example is shown in Figure 2.1.

```
Given:
  Attributes:
      X Domain {0,1}, Y Domain {0,1}
  Facts:
      F = { X=0, Y=0 ::> Class0
            X=0, Y=1 ::> Class0
            X=1, Y=1 ::> Class1
  Initial Assertion: Null
Output: A description of Class0 and Class1
             [X=0] ::> [Class = 0]
             [X=1] ::> [Class = 1]  (Hypothesis 1)
```

Figure 2.1 - Learning from Examples

The facts, or examples composing the set F, are stated in terms of attribute-value vectors in Figure 2.1. In this type of representation an example is a vector of n terms where n is the number of attributes. This is the type of representation that will be used throughout this thesis. The set of attributes and their possible values, or domains, constitute the representation space. Another description could also have been generated in Figure 2.1 for class1. This more complex description is:

[X=1][Y=1] ::> [Class =1] (Hypothesis 2)

Both Hypothesis1 and Hypothesis 2 describe all of the examples in F (and are thus complete with respect to F) and both cover only those examples from Class 1 and not Class 0. In larger problems the number of equivalent hypothesis can be very large. A heuristic, or bias for selecting one hypothesis from a set can be provided explicitly as part of the background knowledge (in the form of a preference criterion in AQ15c) or implicitly based on the learning algorithm used. Such a selection is sometimes known as a learning bias (Gordon and Desjardins, 1995)[i].

2.1.2 Representation Space

The *representation space* is the space in which fact (examples), hypotheses and background knowledge are represented. The representation space is spanned over descriptors that are elementary concepts used to characterize examples from some viewpoint. Individual cells in the representation space correspond to individual examples and are defined as vectors of single argument descriptors (attributes). The *hypothesis language* is the language used to describe concepts within the representation space. Typical constructs of the hypothesis language include nested axis-parallel hyper-rectangles (decision trees), arbitrary axis-parallel hyper-rectangles (conjunctive rules with internal disjunction, as used in VL_1), or hyperplanes or higher degree surfaces (neural nets). The hypothesis language used in this thesis is VL_1 although some of the techniques could be used with any hypothesis language (and associated learning algorithm).

2.1.3 Constructive Induction

Constructive induction performs two intertwined searches: a search for an adequate representation space and a search for hypotheses within that space. The problem of representation space search in an attribute-value representation can be formally defined as:

Given:

* A set of attributes, A
* Background knowledge that defines constraints imposed on the the attributes in A
* One or more of the following:
    - Knowledge defining relationships between attributes in A, or about individual attributes in A
    - A set of observational statements (facts), F, that represent a collection of implications where each statement denotes a description of an example of concept

[i] The approach used by one method for building more robust learning algorithms is to better understand and control all formns of learning bias. This approach is described in Section 2.3.

- A tentative inductive assertion

* Background knowledge that defines preference criterion characterizing the desirable properties of the sought representation space

Find:

* A representation space satisfying the preference criterion.

The preference criterion for a new representation space may consist of statistical metrics describing the minimal ‘information value’ of an individual attribute, or the minimal correlation between an attribute value range and the class. It may also be indirectly evaluated by testing the predictive accuracy and simplicity of the hypotheses learned from that space.

When background knowledge about the problem is used to find an improved representation space, the search is knowledge-driven, and an algorithm which combines this knowledge-based search with a search for a hypotheses is known as knowledge-driven constructive induction (KCI) (Wnek and Michalski, 1994). Similarly, when a search for hypotheses is combined with an analysis of the given set of examples is to guide the search for an improved representation space, it is known as data-driven constructive induction (DCI). Hypothesis-driven constructive induction (HCI) occurs when a hypothesis (which may be learned from a subset of the examples in a previous iteration) is used for the representation-space search in a CI learning system.

2.2 Assumptions of Selective Inductive Learning Algorithms

Before discussing how a more general inductive learning algorithm could be created, it is useful to analyze in more detail the assumptions made by selective inductive learning methods because it can predict when these methods will fail and what can be done to overcome this failure. These assumptions are true for all selective induction learning algorithms, regardless of hypothesis language or computational strategy. In addition to these assumptions, all learning algorithms, regardless of the large space of possibilities, must make certain choices about which parts of the space to be searched, or which hypotheses are preferable.

Selective induction methods make a number of assumptions about the representation, and the distribution of examples. These assumptions can be categorized into three types: 1) completeness, 2) correctness and 3) appropriateness (Figure 2.2). Completeness refers to the extent to which the knowledge provided to the system is sufficient for the system to generate a complete and consistent description of the various classes. Incompleteness is often unavoidable due to the difficulty in obtaining labelled examples. In a medical diagnosis domain patients displaying all possible symptoms and characteristics will not normally be available. Correctness refers to the accuracy with which data is given to the system. Incorrectness can manifest itself in individual attribute values, attributes themselves or example class membership. A common cause of incorrectness is measurement error. Appropriateness refers to the match between proximity in the representation space and proximity of class membership. Selective induction methods assume that the given data is in an appropriate form so that examples which are close to each other in the representation space are also close to (in the case of hierarchically, or linearly ordered classes) or identical (in the case of nominally ordered classes) in class membership as well (Belyaev, 1991, Rendell and Seshu, 1990). When any of these assumptions are violated the representation space is inadequate for selective induction and poor hypotheses (low predictive accuracy and high complexity) result. To achieve a learning method which can overcome any of these deficiencies of the representation space, methods for correcting each of these inadequacies must be developed.

Although related work on overcoming problems of incompleteness and incorrectness are detailed in the following sections this thesis focuses on only inappropriateness. Problems of inappropriateness like continuous attributes, dependencies between non-class attributes and irrelevant attributes are problems of the representation space. Constructive induction methods, including the MCI method described here, are methods for searching for improved representation spaces. Additional methods for overcoming incompleteness and incorrectness may be added to an MCI system, but are out of the scope of this thesis.

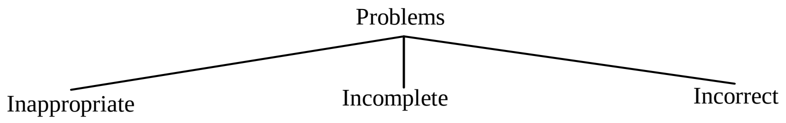


Figure 2.2 - A Taxonomy of Representation Space Inadequacies

2.2.1 Inappropriateness

An induction problem is inappropriate to a representation language if there is a mismatch between the concept boundaries in the space and the capabilities of the descriptive constructs of the language to describe these boundaries. The source of this inappropriateness can lie in the set of attribute values, or the attributes themselves.

2.2.1.1 Abstraction

An example of inappropriate attribute value set would be one in which the provided values blur the concept boundaries by being too broad or too precise. Value sets that contain too few values can be difficult to learn discriminatory rules from because the granularity is too coarse. One approach for handling this problem is to increase the granularity. Such an approach would be performing a concretion operation (in terms of the Inferential Theory of Learning (Michalski, 1993)) and would require knowledge from some outside source.

An attribute domain that contains a large number of values (as occurs with continuous attributes especially) can also cause problems. Many induction methods, such as decision trees and decision rules, perform best when value sets are small and appropriate to the problem at hand (A demonstration of this is given in section 5.1). The size of an attribute domain can sometimes be a measure of the level of granularity of an attribute: a large attribute domain means that examples are precisely defined along that dimension and vice versa. Over-precision can result in learned descriptions that are too precise and overfit the data. Overprecision in attribute value sets is sometimes difficult to avoid when the data provided to the system is continuous, and meaningful discretization intervals are unknown. Although they are performing an abstraction of the given values, this work is usually referred to as automatic discretization of attribute data (Catlett, 1991; Pfahringer, 1995; Fayyad and Irani, 1993), or as a method for efficiently splitting on continuous attributes (Fulton, et al, 1995). Dougherty et al (1995) proposes a taxonomy of discretization methods based on three different axes: global vs. local, supervised vs. unsupervised and static vs. dynamic. The global vs. local axis refers to the scope of discretization, whether a small part of the space is discretized in the context of other attributes as in c4.5 (Quinlan, 1996) or the entire space is discretized over each attribute independently. The supervised vs. unsupervised axis refers to the use of class information when making decision about appropriate interval boundaries, and static-dynamic describes whether discretization is performed for all attributes simultaneously or in sequence. The Chi-merge algorithm (Kerber, 1992) adapted for use within AQ17-MCI is a supervised, global static discretization method which iteratively merges adjacent intervals of an attribute until a χ^2 threshold set by the user is met.

2.2.1.2 Attribute Independence

Inappropriate attributes are those attributes which are relevant to the problem at hand, but which are not independent of each other. Because selective induction algorithms assume that the dimensions of the sapce can be searched independently, such cross-attribute concepts are difficult to capture. For example, the parity problem when stated in terms of the presence or absence of individual attributes, is an attribute-inappropriately stated problem for any induction method which uses axis-parallel hyperrectangles as descriptive constructs. When inappropriate attributes exist attribute construction methods can be invoked which try to combine the given attributes in more problem-relevant manner. A number of systems have been developed with this goal. These methods can be classified into data-driven, hypothesis-driven, knowledge-driven and multistrategy (Wnek and Michalski, 1994; See also Section 2.3.3). Some representative of each of these types are: AQ17-DCI (Bloedorn and Michalski, 1991, Bloedorn and Michalski, 1996) BLIP (Wrobel, 1989), CITRE (Matheus, 1990), FRINGE (Pagallo, 1989), MIRO (Drastal, 1989) and STABB (Utgoff, 1986). The AQ17-MCI system uses methods from AQ17-DCI, and AQ17-HCI (Wnek and Michalski, 1994).

2.2.1.3 Irrelevant Attributes

An attribute can also be inappropriate if it is not relevant to the given classification task. Selective induction learning methods perform a selection of attributes from the given set, and as such are not significantly affected by small numbers of irrelevant attributes. However, with an increasing need to automate the process of knowledge discovery from data, and to find patterns as quickly as possible, induction methods are needed which are robust to even large numbers of irrelevant attributes.

Work in detecting and removing irrelevant attributes can be divided into a filter approach or a wrapper approach (John, Kohavi and Pfleger, 1994). In the filter approach feature selection is performed as a pre-processing step to induction. Because it is separated from the induction algorithm filters are fast, they can be used with any induction algorithms once filtering is done, and can be used on large datasets. However, they they may not agree on the relevancy of certain attributes with the induction algorithm (Imam, 1996). Methods for filtering include those based on information theory as in DCI-SEL, included in AQ17-MCI (Section 4.5.3.1), and the method of Koller and Sahami (1996), LVF, a probabilistic method (Liu and Setiono, 1996), and RELIEF (Kira and Rendell, 1992) which uses heuristics and samples of data in order to find relevant attributes (Liu and Setiono, 1996). The wrapper approach uses the induction algorithm itself to make estimates of the relevance of the given attributes. This can also be called hypothesis-driven filtering. This is the approach taken in HCI-SEL included in AQ17-MCI (See Section 4.5.3.2), and also (John, Kohavi and Pfleger, 1994; Vafaie and Dejong, 1994).

2.2.2 Incompleteness

An induction problem is incomplete if attribute values, attributes (descriptors), or examples are missing. Incompleteness with respect to examples is a fundamental problem in all but trivial summative induction cases. Thus, although selective induction methods do not assume a complete set of examples will be available for learning, they do assume that the training set has a certain degree of completeness. This degree of completeness is satisfied when the training set contains a sufficient number of representative, or prototypical examples so that class boundaries can be accurately determined. Example incompleteness is addressed in knowledge acquisition systems such as DISCIPLE (Tecuci and Kodratoff, 1990). In DISCIPLE, machine learning methods are used to guide the questioning of the expert to most efficiently fill gaps in the knowledge base. Selective, or constructive induction methods do not have mechanisms for acquiring new examples.

A problem may also be incomplete due to attribute (concept) incompleteness. Attribute incompleteness is present when identical examples are present in more than one class (when ambiguous examples exist in the data). This type of incompleteness is a common problem. Methods for overcoming attribute incompleteness differ from those designed for overcoming inappropriate attributes because in this case the attribute required is not a simple function of the current attributes. Thus methods which try combinations of current attributes will fail here. The source for these attributes must come from a domain expert. This knowledge acquisition task is most effective when it is focused on filling the known gaps in the knowledge base. One method for overcoming this problem is CERI which is a part of NeoDICISPLE (Tecuci, 1992). In this approach similar to the repertory grids based on personal construct theory of Kelly (Kelly, 1955), a guided interaction with the user takes place in which the expert is asked to make distinctions between concepts appearing in the positive and negative instances of the rule. In asking very specific questions to the user, the elicitation of useful knowledge is easy for the expert and yet is useful for filling in 'gaps' in the knowledge base.

2.2.3 Incorrectness

Problem incorrectness occurs when some attribute-values, attributes or instances are incorrectly labelled. Incorrectness or error can occur in any stage along the data acquisition process. Incorrectness is most often associated with noise in the training data due to poor sensor readings. Differentiating between the effects of instance noise (misclassification) and attribute-value noise is extremely difficult as often the only manifestation of this noise is the distribution of exceptional instances which are distant from other instances of the same class in the representation space.

Some methods for dealing with incorrect instances, or attribute values, are based on identifying noisy or exceptional instances by using statistical methods applied to the distribution of attribute values, or instances such as in ESEL (Michalski, and Larson, 1978). An alternative method is to use rule coverage to guide example selection as in AQ-NT (Bala, 1993).

2.3 Methods for Building more General Learning Algorithms

As has been shown in Shaffer (1994) and Rao, Gordon, and Spears, (1995) there is no hope for building a universal learner. However, it may be that there do exist learning methods for a large set of the more interesting possible concepts to be learned. In order to find an inductive learning algorithm that can perform well on a wide variety of problems researchers have taken three general approaches: 1) explicitly delineate the biases used by the learning algorithm and select the best for the given learning problem 2) select the best a priori defined bias by selecting from a fixed set of learning algorithms, or 3) modify the representation space[ii].

2.3.1 Modify the Inductive Bias

Because there exist many equally consistent inductive generalizations of a given set of training examples, additional knowledge is used to bias the search toward the most preferred generalizations, or parts of the representation space (Gordon and Desjardins, 1995); (Mitchell, 1980). If the correct bias for a learning problem is selected, then the correct generalizations are made resulting in increased predictive accuracy, reduced complexity and reduced learning time (Provost, 1992). Unfortunately, it is impossible to know for certain which bias is correct a priori. However, it may be possible to find certain problems for which certain biases perform well. Building explicit controls to various type of bias is the goal of work in this category.

SBS (Provost, 1992) makes explicit a number of learning biases, including the definitions of the space, the method for searching the space. It also introduces the idea of inductive policy. The definition of the space is the hypothesis language (decision trees, decision rules etc.) or the use of attribute-value representation. The method for searching spaces is the way in which the algorithm searches for hypotheses and the heuristics it uses for selecting certain examples from which to learn. Inductive policy is a statement of the reason certain bias choices are made. With explicit control of inductive policy a user can trade-off performance traits such as increased learning speed for decreased coverage of examples.

2.3.2 Select the best Inductive Learner

Another way to build a learning system that performs well on a wide variety of problems is to

[ii]Another way to improve the performance of an inductive learning algorithm is to focus on improving the interpretation of learned hypothses. Michalski's second tier approach is one such method (Michalski, 1989). See also (Kubat 1996).

select the most appropriate learning algorithm for the problem. Simply stated, if different learning algorithms have different strengths, then a combination of these algorithms will have the union of these strengths. This approach is taken by the MCS system (Brodley, 1993) One problem with this approach is finding a set of individual algorithms which span the set of possible learing problems. If this set is too small, then the combination approach is still incomplete. If the set is too large then the selection of the appropriate tool becomes complex and time consuming. The Machine Learning Toolbox (Graner, Sharma, Sleeman, et al., 1992) is an example of this approach in which the user must determine the best learner to use.

The VBMS system (Variable Bias Management System) (Rendell, Seshu and Tcheng, 1987) tries to find a match between problem characteristics and learning algorithms. Initially the meta-level learner has no knowledge of which algorithms are best for which problems. Gradually this knowledge is built up as it tries all its available algorithms and then records their performance. VBMS uses a clustering algorithm, PLS1 (Rendell, 1983), to build descriptions of the regions of expertise of the different learning algorithms. The VBMS system is of particular interest because it tries to perform a meta-level learning. However, it uses only a small set of problem characteristics to guide meta-learning (the number of examples and the number of features) and combines only three similar learning algorithms. Despite these limitations initial results were promising for learning meta-level control.

Further motivation for learning characterizations of learning algorithms in hopes of better predicting which algorithm will perform best for a given problem comes from Aha (1992). In this work Aha describes an empirical method for generalizing results from case studies and lists characteristics of problems from which to learn these rules. In this method he proposes a five step approach, the first step of which is to collect case study details. These meta-level attributes describing the problem are, unfortunately difficult to obtain without detailed knowledge of the target concepts. These difficult attributes include: "number of target concepts", "correlations of attributes to target concept disjuncts", "distribution of instances within disjuncts of target concepts", "distribution of instances among concepts" and "amount and type of noise subjected to the instances, attributes and target concepts". The meta-attributes of AQ17-MCI listed in Table 4.1 do not require such detailed knowledge of the target concepts.

The work (Brazdil et al, 1994) is also interested in characterizing learning algorithms. In this work the results of the StatLog project (Michie, et al 1994) were used to build characterizations using C4.5. The StatLog project includes results for 22 different learning algorithms on more than 20 different datasets. Classification problems are described by 15 meta-level attributes. These include 'simple' measures, statistical measures and information-based measures (Figure 3.2). The simple measures are

The number of examples (N),

the number of attributes (p),

the number of classes (k)

the proportion of binary attributes (Bin_att),

the errors quantified by costs (Cost)

the statistical measures include

standard deviation ratio (SDratio)

mean value of correlation (Correl)

canonical correlation for the best single combination of attributes (Cancor1)

the first normalized eigenvalues of canonical discriminant matrix (Fract1)

skewness

kurtosis

The information-based measures include:

entropy of classes (Hc)

entropy of attributes (Ha)

mean mutual information of class and attributes (Mca) noise-signal ratio (Ha-Mca)/Mca

The goal of this work differed from that of meta-rules of AQ17-MCI not only in that it was learning rules for different learning algorithms rather than CI operators, but also in that it was trying to predict the numerical value of the error. AQ17-MCI meta-rules are only intended to predict which operator will perform best.

2.3.3 Modify the Representation Space

A different approach to solving the problem of selective superiority is to fix the inductive algorithm and its bias, and to search for an improved representation space. This is the constructive induction approach as described in Section 1.3. Here constructive induction is viewed as any method that modifies the representation space. Changes may be expansions to the representation space caused by attribute construction or attribute-value addition, or they may be contractions caused by attribute removal or attribute-value removal.

The idea of constructive induction was first introduced by Michalski (Michalski, 1978). Since then a number of other methods for constructive induction have been developed. In (Wnek and Michalski, 1994) a classification of these approaches is introduced based on the primary method used to guide modifications to the representation space. The four methods include knowledge-driven (KCI), data-driven (DCI), hypothesis driven (HCI) and Multistrategy (MCI). The following sections will describe each of the methods in more detail.

2.3.3.1 KCI

Knowledge driven methods base representation space modification on knowledge provided by the user. This knowledge is usually represented as definitions of domain-specific transformations such as constructing a new attribute. Such domain-specific knowledge often results in useful new attributes and shorter more predictively accurate rules. However, such knowledge is often hard to obtain or sometimes express.

Donoho and Rendell (1996) identify an ordering of knowledge used for constructive induction ranging from complete theories to fragmentary knowledge. At one end of the scale is the MIRO system (Drastal et al, 1989) which uses an almost complete theory from the domain to build an abstraction space in which learning will be better, LAIR (Elio and Watanabe, 1991), which is an incremental deductive approach to attribute construction, and TGCI (Donoho and Rendell, 1995). The AQ15 and AQ15c programs (Michalski et al, 1986; Wnek et al, 1995) requires complete knowledge when making use of its a-rules, l-rules or b-rules tables for defining new attributes. At the other end of the scale are systems which use fragments of knowledge. Knowledge of attribute units (dimensional analysis) is used in the COPER system (Kokar, 1986) to guide the system toward meaningful combintions of attributes, and in AQ17-DCI (Bloedorn and Michalski, 1996; See also Section 4.5.5.4).

2.3.3.2 DCI

Data-driven methods base representation space modifications on an analysis of the provided training data. The correlations and interrelationships between attributes are used to expand the representation space by constructing new features, or possibly reducing the space by removing irrelevant attributes. Data-driven approaches are appealing because they do not require domain expertise, and are not tied to a specific knowledge representation such as trees or rules.

The GALA system (Hu and Kibler, 1996) especially emphasizes the ability of data-driven approaches to be used as a pre-processor for a variety of learning methods. The authors demonstrate the effectiveness of GALA's construction of new attributes for C4.5, CN2, a perceptron and backpropagation. GALA also introduces a useful concept of relative improvement for judging the quality of new attributes. Because such attributes are generated from those given it would be unnecessary to introduce a new attribute, even if it had a high absolute 'value', if it did not help dicriminate the given classes any better than its parents. GALA is limited however in it approach in that it performs only feature construction, uses only boolean operators of 'and' and 'not' and requires that the given attributes be converted to booleans before processing. The step of 'booleanizing' adds is expensive: the authors report a complexity of O(AVE) where A is the number of attributes, V is the maximum number of attribute values and E is the number of examples.

AQ17-DCI (Bloedorn and Michalski, 1991, 1996), expands the representation space through attribute construction, and contracts the space through attribute-value abstraction and attribute removal. Attribute construction is based on a generate and test approach using algebraic and logical operands. Using the set of operators selected by the user the algorithm generates pair-wise and multi-argument functions of the original attributes. These attributes are evaluated using an information gain ratio (Quinlan, 1989), or PROMISE (Baim, 1982) metric. If their information value is greater than a user-defined threshold, then the new attribute is added to the available set for learning. Abstraction of attribute values is performed using the Chi-merge algorithm (Kerber, 1992). This algorithm repeatedly merges adjacent intervals while the merging does not overly blur class boundaries. Attribute-value and class correlation is calculated using a χ^2 statistic. Attribute selection is based on a simple filter - those attributes with an individual information value less than a user-defined threshold are removed from the available set.

Another method for attribute construction from data is done in BACON (Langley, Bradshaw and Simon, 1983; Langley et al, 1987) and ABACUS (Greene, 1988). These programs do not search for class descriptions given labelled examples, but instead build new attributes that are numerical functions of the original attributes. Attribute construction is based on the interdependencies between attributes in the data. EF, the equation discovery function within FAHRENHEIT is another equation discovery algorithm (Zembowicz, and Zytkow, 1991). Unlike BACON, however, EF takes a more rigorous approach to handling the problem of propagating error when generating new attributes which allows it to converge faster to equations which better match the data. These approaches are closely related to the work of Sutton and Matheus in their program for finding polynomial functions (Sutton and Matheus, 1989) and of canonical discriminant analysis in CAF (Yip, and Webb, 1994). The former work uses a linear regression algorithm is used to find candidate attributes. The pairs of attributes found are used as a basis for new attributes. The CAF algorithm uses discriminant analysis to find a linear combination of discriminating attributes. CAF performs a a statistical clustering analysis to find new attributes which maximally separate class centroids.

STAGGER (Schlimmer, 1987), uses a variety of data-driven methods to search for a good representation and to find the best hypotheses within the representation. These three interacting components include 1) a module for adjusting the weights of symbolic descriptions, 2) a module for adding new boolean combinations of features, and 3) a module for abstracting attribute-values into discrete intervals. Attribute construction is only invoked when STAGGER makes a prediction error on a training example. The type of construction made is based on the type of error. When the hypotheses is too general a new element which specializes the current hypotheses is added using AND. When the hypothesis is too specific, a new element which generalizes the hypotheses is added using OR. The attribute-value abstraction module uses a beam search to quantize a real-values range into discrete intervals. Each potential interval is merged or retained based on the number of positive and negative examples, or utility of that interval. STAGGER more tightly couples learning with representation space modification than AQ17-MCI, but it has fewer operators for representation space change, and does not learn control knowledge that will carry over between learning problems.

LFC (Ragavan and Rendell, 1993) performs data-driven construction of new attributes while constructing a decision tree. LFC uses a lookahead before it creates each node in the decision tree. This lookahead is constrained by a number of heuristics in order to allow LFC to be of reasonable speed. A benefit of this lookahead is that the replication problem common in decision trees when the available examples are difficult to split on one attribute, is much reduced. Like STAGGER, LFC uses only logical operators to construct new terms from the original set. MRP (Perez and Rendell, 1995) also performs feature construction iteratively with decision tree generation. Unlike LFC, however, MRP uses multidimensional projections to find arbitarily complex logical relationships. Although the projection method used within MRP has also been used to perform feature selection, MRP is primarily designed for feature construction. Both MRP and LFC, primarily differ from AQ17-MCI in the way they generate new attributes while generating a decision tree, in their use of only methods for attribute construction and not selection or abstraction (and as such have no meta-level of control for representation space modification), and in their restriction to only logical combinations of attributes.

2.3.3.3 HCI

Hypothesis-driven methods base modification of the representation space on patterns found in the generated hypotheses. These patterns can signal the method that new attributes should be constructed or removed. Hypothesis driven methods do not require domain expertise, but are tied to a specific hypothesis language.

Utgoff's STABB (Shift to a Better Bias) program constructs new features to add to the concept description language when a new example is not properly classified by the current hypotheses (Utgoff, 1986). New features are built using a least-disjunction method or a constraint backpropogation method. In the least-disjunction method the least-specific disjunction of existing terms is built which corrects the current hypothesis. In the constraint backpropogation method new terms are built deductively by tracing the reasoning used when applying a successful operator. These operators control problem solving behavior in the LEX system.

AQ17-HCI (Wnek, and Michalski, 1994) can both expand the representation space by building new attributes based on strong patterns found in the learned hypotheses, and contract the space by removing attributes. Attributes are built by detecting the best-performing patterns found in a previous iteration. Patterns may be a group of rules, a part of a rule or even a part of the condition of one term. The strength of a pattern is a function of the number of positive and negative examples covered by the pattern.

FRINGE (Pagallo and Haussler, 1990) is a hypothesis-driven method for constructing new attributes based on patterns found in decision trees (FRINGE). The FRINGE algorithm generates new features by conjoining or disjoining the parent and grandparent nodes of all positive fringes in the tree depending on the position of the child node relative to the parent. This algorithm was found not to perform well on CNF-type problems which led to the introduction of Symmetric FRINGE (Pagallo, 1990) and DCFringe (Yang, et al , 1991). Despite these modifications these algorithms appear to be useful only for a small class of DNF and CNF-type concepts.

CITRE (Constructive Induction on Decision Trees), (Matheus, 1989) generates new conjunctive terms after a decision tree is built. Feature construction is motivated by excessive disjuncts (replication) in the learned hypotheses. New terms are constructed based on attributes found along the tree path (recent path, or all the way from root to leaf). These new features are filtered out based on possible domain knowledge, and evaluated using either an information gain metric, or a competitive metric. In the latter, the usefulness of a new feature is based on whether it is used in later decision tree construction.

2.3.3.4 Multistrategy Constructive Induction

Multistrategy constructive induction methods use a combination of the KCI, DCI and HCI approaches to make changes to the representation space. Multistrategy methods are thus the most flexible approach, but also require additional heuristics of computation in order to select and integrate the other approaches.

CN2-MCI (Kramer, 1994) uses both hypotheses and data to guide attribute construction. CN2-MCI first analyzes the learned hypotheses to find co-occurring attributes. These pairs of attributes are selected as the operands for new attributes. The examples in the two-dimensional projection of these attribute pairs are then clustered to find the definition of the new attribute. Only the top new attributes are retained for the next iteration of learning and evaluation. The cycle of learning, evaluation and attribute construction stops when there are a user-defined number of iterations with no improvement. CN2-MCI is categorized as a multistrategy constructive induction method because it uses information from both hypotheses, and data to generate new attributes. However, it differs significantly from AQ17-MCI in that it does not include other methods for representation space modification including representation space contraction by attribute removal or abstraction. Additionally, because AQ17-MCI performs multiple transformations to the representation, it includes control strategies for selecting the most appropriate one for the given problem. This meta-level control of representation space modifiers is not included in CN2-MCI because it only performs feature construction.

CHAPTER 3 MULTISTRATEGY CONSTRUCTIVE INDUCTION

3.1 Background

Related work in the area of constructive induction, and multistrategy constructive induction was given in the previous chapter. This chapter describes and motivates the novel and more general architecture found in AQ17-MCI. Multistrategy constructive induction is a process for learning hypotheses from examples in which the search for the preferred hypotheses is combined with a multi-operator search for an improved representation space. This more sophisticated approach is motivated by the inability of simpler methods to perform well on problems that contain one or more of the following pathologies: a) large numbers of irrelevant attributes, b) large, possibly continuous attribute-value domains, and c) attributes with interdependencies. The goal of this research is to not only produce a single learning system which can solve problems made difficult by these pathologies individually, but to produce single system which can learn even when two or more of these pathologies are present. To achieve the level of performance needed for the increasing number of real-world applications where such patholgies are common, a multistrategy method for learning is required.

To design an effective multistrategy architecture each of the components of a multistrategy system must be carefully selected. This chapter describes the design decisions involved in building a multistrategy constructive induction system, introduces and motivates a novel learned approach to MCI, and describe the links between this design and the inferential theory of learning (Michalski, 1993, Michalski, 1992).

3.2 Design Goals

The goal of this multistrategy constructive induction (MCI) learning method is to build a learning system that can generate simple, predictively accurate hypotheses from a learning problem that is poorly represented. The representation space may be poor due to the presence of irrelevant attributes, too much detail in the attribute values, or inter-dependencies between attributes. Some representational problems, however, such as incompleteness (no relevant attributes, or attribute-values present), are outside the scope of this thesis. This goal requires the system to be able to perform a wide variety of transformations on the input provided. Clearly, there are two factors important in controlling how successful such systems are in achieving this goal: 1) what transformation operators are combined and 2) how they are combined.

3.3 Selecting Representation Space Modification Operators

The choice of individual representation space modification (RSM) operators available to a multistrategy approach determines the capabilities of the whole system. The individual strategies of a multistrategy system should be selected for their coverage of the possible types of problems. A set of operators should be selected so that the resulting whole will efficiently solve most, if not all, of the types of problems that could arise. In chapter 2, three categories of learning problem pathologies - inappropriateness, incompleteness and incorrectness were described. AQ17-MCI includes operators for representation space expansion (e.g. attribute construction) to overcome inappropriateness (of attributes), and representation space contraction to overcome some incorrectness and inapropriateness (of feature values). While a large set of operators may appear best in order to overcome the greatest number of pathologies, a minimal set of such strategies reduces the chance of redundancy and makes operator selection simpler.

A strong set of RSM operators is of little use, however if they cannot be used together due to differences in representation. Strategies which represent knowledge in different forms (e.g. decision trees and connectionist networks) may require the construction of translation methods or knowledge interchange format so that such methods can communicate results to each other and to the final hypothese language. Such translations often result in loss of information and will require additional computation time and resources. A preferred approach would have a set of operators that operated on the same representation.

The best design of an multistrategy constructive induction system is one that includes a wide variety of disjoint methods that can perform both expansion and contraction of the representation space, and which share a common representation.

3.4 Controlling Representation Space Modification Operators

Another factor in determining the success of a multistrategy approach is the method of strategy selection and combination. Just as learning can be characterized as a search through the representation space, the selection of an appropriate operator in a multistrategy system can be characterized as a search through a space of operator choices. For this reason well known search methods can also be used as operator selection methods. A description of the relative strengths and weaknesses of different search operators when applied to this problem follow.

3.4.1 Random and Exhaustive Operator Selection

The simplest method for combining two or more operators is to randomly select between them. This method may sometimes find the correct ordering, but it may take an unnecessarily long time by selecting operators that are clearly not needed (e.g. if it selects attribute abstraction for a multiplexor problem), or it may even perform operations that user does not wish (e.g. the user has a preference for DCI over HCI-based attribute generation). When no other alternative is possible random selection can be the only alternative. Another simple approach is an exhaustive or parallel search in which all possible combinations are tried and evaluated. This method is guaranteed to find the best possible solution, but it will be the most expensive possible method in terms of time and memory and may be compuationally infeasible for even moderately sized problems. A variation of this exhaustive method is to perform a sampling evaluation. In a sampling method only a subset of the data, repeatedly sampled, is used to evaluate the usefulness of operator transformations on the problem. The operator selected for the problem is the one with the best average performance over all the samples. The greatest drawback of this method is that it will still require a relatively large amount of time and resources to have enough certainty in the results. It is also unnecessary to sample every possible operator when the characteristics of the problem can offer some guidance. For example feature selection is not necessary if the number of attributes is already small. The algorithmic approach described next partially addresses this by recognizing that the operators have different characteristics and may be optimally ordered.

3.4.2 Algorithmic Operator Selection

Another class of search methods uses an iterative, but fixed approach. In this algorithmic class the available search operators are applied one at time in a pre-specified order. For example in a constructive induction problem the system can be set to always reduce the space using attribute-removal and attribute-value abstraction before doing any attribute construction. In this way it is hoped the algorithm will be faster because attributes which are irrelevant will not be combined. However, this approach may prove counter-productive because attributes which are useful in combination may appear to be irrelevant alone. Consider the simple example shown in figure 3.1. In this two-class classification problem x1 and x2 individually are uncorrelated with class. However, it is clear to see that x1=x2 for class1 and x1<>x2 for class2. Such a simple combination could not be found if either these attributes were removed. Similar problems occur in every fixed algorithmic ordering of operators. For example, abstraction followed by attribute generation can result in much different rules than attribute generation followed by abstraction. In Section 5.4.3 MCI is applied to a computer vision problem. In this problem DCI-Quant (abstraction) followed by DCI-Gen (generation) resulted in hypotheses which on average had 35.8 rules, 679.1 selectors and took 83.1 seconds to learn. With the ordering reversed the hypotheses contained only an average of 3.3 rules, 54 selectors and took only 3 seconds to learn. However, the opposite ordering was better in the application of MCI to economics. In this problem described in Section 5.4.2. Here, abstraction followed by generation resulted in rules with an average predictive accuracy of 76.3%, the opposite ordering resulted in hypotheses with an average predictive accuracy of 43.7% accurate. The ordering is optimal for one class of problems, but not another. This still fails to achieve the goal of a widely applicable learning approach. The other problem with this approach is that it cannot take advantage of user-expertise or domain knowledge when it is available, and can not improve its performance over time. It will always perform the same for a specific learning problem, regardless of past experience.

class1		class2	
x1	x2	x1	x2
2	2	2	3
4	4	3	4
3	3	4	2

Figure 3.1 - Example of individually irrelevant attributes which provide perfect classification in combination

3.4.3 Heuristic

Another approach to finding an improved representation is to allow the user to either make the changes to the representation space directly through knowledge-driven CI (the user removes attributes, adds new attributes, or selects new attribute-domains) or to control the representation space modification tools available in AQ17-MCI. Both KCI, and direct user-control of operators is supported in AQ17-MCI. The drawback of both approaches is that it relies on the expertise or domain knowledge of the human controller to either to make the correct changes to the representation space, or to select the correct sequence of operators to solve the problem. This toolbox approach simply makes program options and parameters accessible to the user. If the user does not understand the problem and the tools well enough, then this approach reduces to either the previous algorithmic, or random approaches described earlier. If the user is knowledgeable about the representation space and which attributes should be removed and/or combined, then a constructive induction learning method is hardly needed. The user has enough expertise to set up the problem to either solve it himself, or allow a simple selective induction learner to solve it. Although it is an approach that allows the user to provide knowledge when it is available (knowledge-driven constructive induction or KCI) it fails when this knowledge is unavailable and it requires the constant attention of the expert or it will repeat past mistakes.

A more sophisticated form of heuristic search in which general, rather than (or in addition to) domain-specific expertise is used, is also possible. In this form of heuristic search operator-rules are used to guide operator selection. These rules map a description of the representation space itself to the choice of operator. For example a operator for removing attributes may only be invoked if the total number of attributes used to describe the given problem is greater than 20. By using general problem characteristics in conditions of these operator-rules this type of heuristic search is more general the specific KCI approach previously described. A weakness of this approach is the difficulty in describing the conditions under which certain operators are successful against when they are not. These rules may also need to be updated as new problems are encountered.

Some work has been done in knowledge acquisition aimed at helping the user construct a knowledge base that could be used for this meta-learning task. The KAISER system (Dabija et al, 1995) is used to iteratively refine, by means of interactions with a domain expert, the knowledge encoded in a decision tree of that expert's domain. One 'impropriety', or problem with a decision tree, that KAISER is designed to correct is 'Similar class'. This is the tree replication problem and occurs when "...more than one node tries to separate the same set of classes at different places in a decision tree." (p. 94). The correction for this problem is to generate a new attribute which alone separates the examples in the subtree. As stated by the authors, the difference between this approach and the constructive induction (CI) approach to repairing poor decision trees is that CI attempts to fix the problems without assistance from the user. However most CI methods do not take into account the wide variety of possible 'improprieties' that a learned hypotheses may contain, and the wide variety of corrections that made be made. The next section outlines a more general CI approach to this problem.

3.4.4 Learned Control Rules

It is from the weaknesses and strengths of the previously described methods for operator selection that a novel approach to operator selection using learned operator-rules was developed. Previous approaches were poor because they could not take advantage of domain specific knowledge when available, and could not easily be updated when a new learning situation found them to be performing poorly. A method which learned control rules for selecting the appropriate representation space modification operator would be useful because it would require little knowledge on the part of the user, but could use it if it was available, and could automatically update its operator selection approach based on experience.

Motivation for learned control rules arise partially from the realization that learning methods are often used on a sequence of learning tasks, and that the completion of previous related tasks can improve the performance of the learner on a new task when these problems are related (Thrun and O'Sullivan, 1996; Caruana, 1993).

Guiding constructive induction via learned control rules is a strong method because it can use user-supplied knowledge when available, and it can learn from past failures and successes.

Learned control rules in the form of decision rules are readily accessible and comprehensible to a domain expert. These rules can be understood and manually revised if needed. An example of such a rule is shown in Figure 3.2. Further motivation for attempting to characterize CI operators is provided by the work of Aha (1992) and and the success of (Brazdil et al, 1994) described earlier in Section 2.3.2.

if [num_attributes = 48..64] ::> perform_data_driven_attribute_selection

Figure 3.2 - An example rule controlling the invocation of data-driven attribute selection

Furthermore, such rules can also be learned empirically from examples of past success and failures. A description of a system which performs this operation is given in chapter 4.

The previous section described the decisions important to building an effective multistrategy constructive induction algorithm. These design considerations are important for the design of any multistrategy learning system. The next section shows how the operations performed by MCI can be understood in terms of fundamental inferential transmutations of the Inferential Theory of Learning (Michalski, 1993). This analysis not only provides a general framework in which to describe the current capabilities, but it also suggests new capabilities of MCI.

3.5 An Analysis of MCI based on the Inferential Theory of Learning

This section dicusses the links between the Inferential Theory of Learning (ITL), to the representation space modification operators (RSMOs) as they are used in MCI, and the architecture of MCI itself. A complete discussion of the Inferential theory is available in (Michalski, 1993), but a brief description is included here for clarity.

3.5.1 Outline of the Inferential Theory of Learning

In the Inferential Theory of Learning (ITL), learning is viewed as a goal-oriented process of improving the learner's knowledge. Learning processes are viewed as patterns of inference called knowledge transmutations. Some of the basic transmutations include generalization, abstraction, and specialization. A basic premise of the theory is that learning processes can be described in terms of the application of various inferential operators. Along one dimension the theory classifies inferential processes into inductive and deductive. Inductive processes are falsity preserving, while deductive processes are truth preserving. To use the fundamental equation for inference: P U BK |= C, where P stands for Premise, BK is the reasoner's background knowledge and C is the consequent, deduction derives C given P and BK, while induction derives P given C and BK. The second dimension of the classification of inferential processes is contingent vs. conclusive. Conclusive processes are strong and are based on domain-independent knowledge, while contingent processes are weak because they are based on domain-dependent knowledge. This classification of inferential transmutations clarifies the differences between many inferential processes and reveals some new, little explored combinations.

3.5.2 Representation Space Modification Operators as Transmutations

This section describes the relationship between ITL transmutations and representation space modification operators (RSMOs). Special attention will be given to those RSMOs in the current MCI architecture, but other possible operators will also be described.

A Multistrategy constructive induction system may include any number of RSMOs. These operators include expanders that increase the size of the space and contactors which reduce the size of the space. Expansion operators include those that add attribute values and those that add attributes. Although the addition of new attribute-values would be a form of concretion and is not presently available in MCI, there do exist many methods for attribute construction. Contraction operators reduce the size of the representation space through attribute removal (or selection) and abstraction.

Expansion operators may make use of a number of different knowledge transmutations. When viewed in the Inferential Theory of Learning, the expansion of the space by the addition of new attributes can be viewed as a *derivation*. transmutation. As described in this theory the underlying process of a derivation may range from a randomization to an equivalence. In attribute construction based on equivalence derivation or reformulation[iii], new attributes are generated from given knowledge structures, (original attributes, generated hypotheses) and are equivalent to their parent structures. This attribute construction method is used in MCI in the #VarEQ(x) operator which counts the number of attributes from a set which have the value x. A specialized case of this attribute construction method is shown in Figure 3.1. The constructed attribute is simply reformulating the three Food attributes into four boolean attributes.

Original attributes:	Food1	Food2	Food3	
	apple	milk	toast	
	milk	eggs	toast	
Generated attributes:	Exists(apple)	Exists(milk)	Exists(toast)	Exists(eggs)
	yes	yes	yes	no
	no	yes	yes	yes

Figure 3.3 - Attribute construction based on an equivalence derivation

Intermediate derivations are neither equivalence relations nor randomizations, and can also be used to generate new attributes. The construction of a new attribute generated by taking the sum of two original attributes would be a deductive derivation. This is the type of attribute construction performed in DCI and also used in AQ17-MCI. Such derivations are important because they make explicit relationships *between* attributes that any inductive learning algorithm

[iii] Matheus (Matheus, 1989) would call this incidental constructive induction because no induction was done during construction

based on an attribute-value representation (e.g. AQ, C4.5, Backprop) is unable to represent. An example of attribute construction based on the deductive derivation operator of comparison is shown in Figure 3.4. Deductive derivations in MCI include: addition, subtraction, multiplication, comparison, maximum, minimum, and average.

Original attributes:	OutputYear1	OutputYear2
	1000	1100
	1121	1221

Generated attributes:	Equal(OY1,OY2)	LessThan(OY1,OY2)	Greater(OY1,OY2)
	yes	no	no
	no	no	yes

Fig 3.4 - Attribute construction based on an deductive derivation (comparison)

Attribute construction can also be based on an analogical derivation. An example of this[iv] can be found in (Bloedorn, 1993a). In this method a genetic algorithm approach is used to search for new attributes using a crossover operator. In ITL a similzation analogy is performed when the "similarity between two entities in terms of some descriptor(s), and the knowledge the one entity has property A" is used to hypothesize new knowledge that the second entity also has property A (p. 23, Michalski, 1993). In the genetic algorithm example we know that parent attributes p1 and

Parent attributes: p1: Property 1) discriminates given classes
Property 2) Is a function of x1 and x2 (e.g. x1+x2)
p2: Property 1) discriminates given classes
Property 2) Is a function of x7, x9 and x5 (e.g. (x7*x9)+x5)

Generated attributes: c1: Property - is a function of x1 and x5 (e.g. x1+x5)
c2: Property - is a function of x7, x9 and x2 (e.g. (x7*x9)+x2)

Figure 3.5 - Attribute construction using analogical derivation

p2 have the property that they are discriminatory attributes. We also know that p1 has the

[iv]The analogical inference described here is more precisely known as a similzation because the reference sets of both objects are in the same class.

property that it is a function of attributes x1 and x2, and that p2 is a function of x7, x9 and x5. It is known that the usefulness of a function in discriminating classes is itself a function of the usefulness of the component attributes. From this we infer that the new attributes (c1 and c2) which share properties of p1 and p2 will also have the propoerty of their parents - that they are discriminatory. An example of the crossover method to construct new attributes is shown in figure 3.5. This type of construction is not currently a part of MCI, but is possible in this framework.

Attribute construction based on inductive derivation requires that the attribute construction method have available meta-level knowledge about the attributes being combined. This is necessary to constrain the search for new attributes. As such, it is very knowledge-intensive and potentially domain specific[v]. However this approach can also produce very powerful new attributes which could predict regions of the space and allow learning algorithms to perform much better with sparse datasets (Matheus, 1990). This type of inductive derivation of new attributes using a goal-driven property transfer algorithm is described in (Bloedorn, 1993b). In an inductive derivation

Parent Attributes:	BayLength+BayWidth	BayLength+BayHeight
Constructed attribute:	BayWidth+BayHeight	

```
IF
   (UNIT Y (IS FT))(UNIT X (IS FT))
   ((MEASURE X) (MEASURE Y) (DIFFERENT X Y))
   (TYPE X (IS NUMERIC)) (TYPE Y (IS NUMERIC)))

THEN
       ADD(X,Y)
```

Figure 3.6 - Attribute construction using inductive derivation

[v] Matheus (Matheus, 1989) would call this region predictive attribute construction

reasoning is performed which derives an explanation from effects. In the example shown in figure 3.6. the effect is that the new attributes (BayLength+BayWidth) and (BayLength+BayHeight) are useful sums (e.g. comprehensible, disciminatory). The explanation for this effect is that these are good attributes to combine because in both cases the original attributes have units of feet, they don't combine the same attribute (BayLength+BayLength) and the type of both attributes is numeric. This explanation leads to the suggestion that BayWidth+BayHeight may be a useful new attribute using the ADD operator. In this example, the system inductively learns a meta-level rule which characterizes when it is appropriate to add two attributes. This new rule is used to construct other attribute combinations.

This new attribute is constructed based on knowledge of the attributes' units, and type. The examples of previous successful attribute constructions shown were inductively generalized to a meta-level rule. This rule was then applied to generate a new attribute BayWidth+BayHeight. This type of attribute construction is not presently available in MCI, but may also be included.

The randomization derivation may also be used to generate new attribute. An example of this is the mutation operator used in the genetic algorithm method for constructing new attributes described earlier (Bloedorn, 1993a). An example is shown in Figure 3.7. In this example the operator used to combine two attributes is randomly reassigned from addition to multiplication.

Parent attributes:	x1+x2
Generated attribute:	x1*x2

Figure 3.7 - Attribute construction using randomization derivation

Contraction operators reduce the size of the representation space through attribute removal and attribute-value removal (or discretization). As it is implemented in MCI, attribute removal is a non-inferential selection transmutation which picks those attributes which are relevant. Selection is a non-inferential transmutation because it does not modify the meaning or content of the input knowledge, but manipulates the attribute values as data (p. 13, Michalski, 1993). Attribute-value removal performs a reduction in the granularity, or “amount of information about a set of entities”. In this case the “set of entities” is the domain of that attribute. Attribute-value removal is performing an abstraction transmutation which in this case (and most typically) is a deductive inference.

MTL Algorithm

1) Use the input to activate the segments of the learner’s prior knowledge base relevant to the input and the learning goal

a) Calculate relevance relationship between input and BK

b) Store knowledge in KB as DIH traces

2) Determine type of relationship between input information to the learning process and the BK

a) The input represents new information - perform synthetic learning to cover new example or store example

b) The input is implied by, or implies a part of the BK - perform analytic learning to derive an explanatory structure which links the input with the involved part of the BK

c) The input contradicts some part of the BK - revise the BK through synthetic learning or manage inconsistency.

d) The input evokes an analogy to a part of BK - if the analogy passes an “importance criterion” store the analogy and its links to the BK

e) The input is already known to the learner - updated a measure of confidence associated with this part of the BK.

MCI Algorithm	
1) Use the input to activate the segments of the learner's prior knowledge base relevant to the input and the learning goal	b) The input is covered by a BK metarule - store the input and update weight of metarule.
a) Calculate relevance relationship between input (meta-vector, result) and BK (stored meta-rules)	c) The input contradicts some part of the BK - specialize the BK metarule through synthetic learning.
b) Store knowledge in KB as rules	d) Not possible in current representation
2) Determine type of relationship between input information to the learning process and the BK	
a) The input represents new information - perform synthetic learning (meta-level) learning to cover new example and store example (full memory model).	e) The input is already known to the learner - updated a measure of confidence associated with this part of the BK.

Figure 3.8. A comparison of MTL framework and the MCI algorithm

3.5.3 MCI as Multistrategy Task-adaptive Learning

The previous section detailed how various representation space modification operators can be viewed as transformations in the ITL framework. Figure 3.8 shows how the overall approach of the MCI method can also be analyzed within the framework of ITL and be shown to be a multistrategy task-adaptive learner (MTL). As described in ITL, (p. 25 (Michalski, 1993)). "The underlying idea of MTL is that a learning strategy should be tailored to the learning task." MCI fits the learning strategy (selecting which RSM operators to use) to the learning task based on the input to the learning process (the problem and its description) and its background knowledge (the learned metarules).

3.5.4 Summary

This analysis of MCI when viewed within the conceptual framework of ITL has provided a deeper understanding of the MCI algorithm. It demonstrates this by showing the mapping between the MCI RSMO operators and the fundamental knowledge transmutations of ITL. Presenting MCI in an ITL framework has revealed additional RSMOs that are not currently a part of MCI. These methods include attribute construction via analogical, inductive or randomization derivations. It also reveals how the framework of the MCI system can be seen as an instantiation of the Multistrategy Task-Adaptive Learning framework.

Another useful benefit of the analysis of MCI in terms of ITL is that it reinforces the need to view learning in a broader scope as goal-guided inference (Michalski and Ram, 1994). This view emphasizes the importance of introspective analysis of the system's knowledge and reasoning processes, and maintaining a memory of past successes and failures. The concept of a goal-dependency network discussed by the authors can also not only be used for reasoning about dependencies and priorities for learning, but also for reasoning about the attributes in the current representation space.

In summary, the MCI method has been shown to perform a double intertwined search for an adequate representation space, and for hypotheses within that space by invoking a series of knowledge transmutations. The transformations performed depend on the input data and the learning goal. The input data consists of the definitions of attributes and their values, the class-labelled examples and the learning parameters. The learning goal is specified at the meta-level by the accuracy and simplicity thresholds of learned rules, and at the learning level by the by the criteria parameters. Although it is currently implemented with a rule-based hypotheses generator, other types of learners using other types of representations could be used. These other representations include decision trees, or connectionist networks. A different set of RSMOs would need to be constructed for non-predicate based representations, but the fundamental architecture of MCI could remain the same. The remainder of this thesis will focus on the current implementation of MCI which uses a decision rule representation for learned hypotheses and learned meta-rules.

CHAPTER 4 THE AQ17-MCI SYSTEM

This chapter describes the design and features of the AQ17-MCI system. First the general architecture is described in section 4.1 followed by detailed descriptions of each of the components in the later sections.

4.1 General Architecture

AQ17-MCI implements the principles of multistrategy constructive induction described in chapter 3. In this approach learning from examples is performed as an iterative double search. One search is for an improved representation space. This search is performed by the Representation Space Modification (RSM) Module. This module includes a toolbox of available operators and a

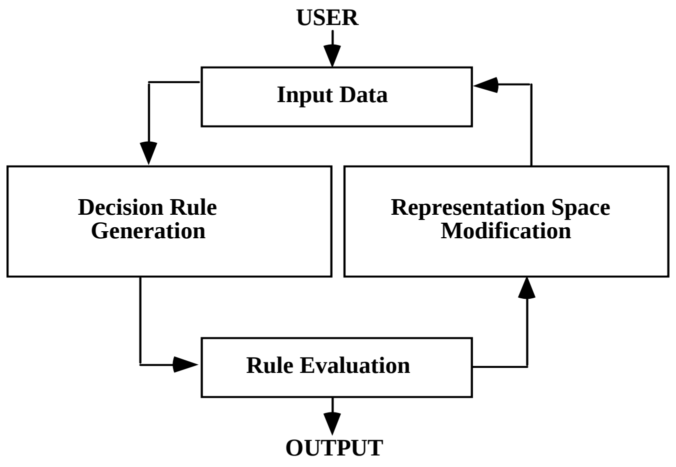


Figure 4.1 - A functional diagram of the MCI method

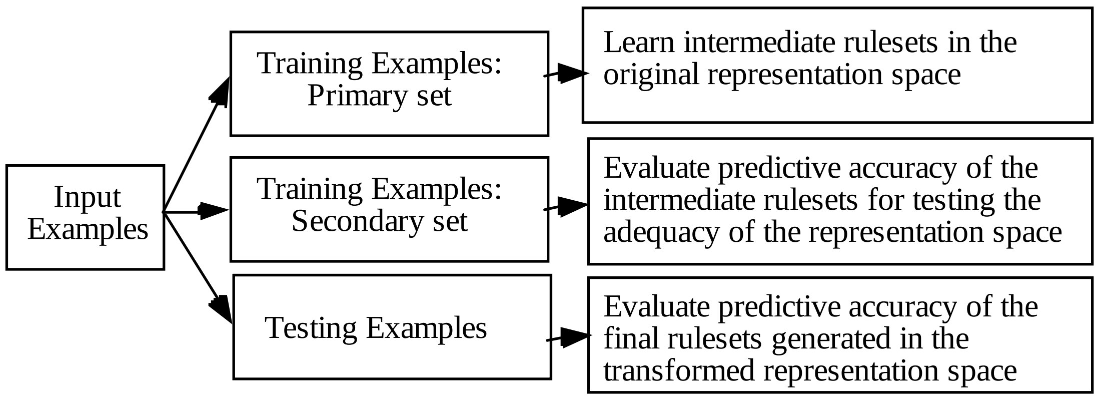
controller for selecting operators for the current problem. The other search is for the best hypothesis within that space, given the preference criteria. This search is performed by the AQ learning algorithm as implemented within AQ15c (Wnek et al, 1995). Control cycles between search for a hypothesis and search for an improved representation space, until the evaluation module determines that either the learned hypothesis meets the user-defined thresholds, or that no more improvement can be made. The general architecture of AQ17-MCI is shown in figure 4.1.[vi]

4.2 Input Data

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 supplied to the Decision Rule Generation module, which uses an empirical inductive learning program (AQ15c) 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 output as the solution), or to move to the Representation Space Modification module. This decision is based on control meta-rules (Section 4.5.). The final decision rules are evaluated on the testing examples to determine their performance. The partitioning of input data is shown in Figure 4.2.

The representation space modification is done by an application of various RSM operators, acting as constructors or destructors. Once a new representation space has been determined, both

[vi]Figure 4.1 shows the decision rule generation module (search for a hypothesis) being invoked before the representation space modification module (search for a representation) as this allows the system to first determine whether modifications to the representation are needed at all.


Input Examples
Training Examples: Primary set
Training Examples: Secondary set
Testing Examples
Learn intermediate rulesets in the original representation space
Evaluate predictive accuracy of the intermediate rulesets for testing the adequacy of the representation space
Evaluate predictive accuracy of the final rulesets generated in the transformed representation space

the primary and secondary training dataset are reformulated into this space, and the process is repeated. The next sections describe in greater detail various aspects of the above process.

Figure 4.2. The partitioning of input examples and their roles

4.3 Decision Rule Generation

The search for hypotheses withing a given representation space is performed in AQ17-MCI by the AQ algorithm. The AQ algorithm, as implemented in AQ15c (Wnek et al, 1995) performs a beam search (the size of which is user determined) for a set of decision rules which cover all the positive examples and none of the negative examples. This search is done by randomly selecting a 'seed' event for a class, covering that example with a very specific cover (conjunct), and then extending that cover in all dimensions, stopping each time a negative example is found. If all positive examples are covered in this one 'star', then a new class is selected and the process is repeated. If not, then a new disjunct is started, and a new seed is selected from the same class and star generation is repeated. This proess of 'seed' selection, and extension-against (negative examples) is repeated until all examples are covered. Afterward the tenative hypotheses may be modified by other post-processes which may generalize the cover by "dropping conditions." The end result is a set of decision rules for each class in the data. An example of a rule produced by AQ is shown below.

```
Class1-Outhypo
# cpx
```

1 [color = blue] [size = small] [shape = square]
2 [color = red]

Figure 4.3 - An example AQ generated rule

This rule states "an object is in class1 if: it is blue, small and square, or it it red". The first line (starting with "1" in Figure 4.3 is a *rule* or complex. A rule is made up of conjunctions of conditions or *selectors* such as [color = red]. AQ15c is a full-featured rule learning program and although it constitutes a part of AQ17-MCI the user has access to all of the learning parameters and options available in AQ15c for controlling such things as rule type and rule intersection.

4.4 Rule Evaluation

Each time new rules are generated they are evaluated. Rule generation occurs either as part of the itial detection step in which the need for representation space change is determined, or after a representation space modification (RSM) operator has been selected and applied to the data. Control is returned to either the representation space modification module, or the process stops depending upon the results of this rule evaluation. Rule evaluation is based on a number of criteria. As described in (Bergadano, Matwin, and Michalski,, 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 AQ17-MCI rule evaluation module uses only accuracy and simplicity to evaluate rule quality. Cost is not used explicitly because it can be included in the preference criteria within AQ to find the rules with minimal cost.

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 consists of those training examples not available to the learner during hypotheses generation. One advantage of this holdout method is that rules learned from the primary training set, but which perform well on the secondary set, are less likely to overfit the original data. Predictive accuracy is measured as the percentage of secondary training examples correctly classified.

The *Complexity* of a ruleset is evaluated by counting the number of rules in the ruleset and the total number of conditions (or selectors).

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.

4.5 Representation Space Modification

The representation space modification module (RSMM) is responsible for determining which modification operator to apply to the current learning problem, recording the operator selected with a vector description of the problem, making the changes to the training and testing examples and updating the meta-rules. Figure 4.4 shows the design of the RSMM.

RSM operator selection is performed by the RSMM by matching the characteristics of the current problem against stored meta-rules. This matching is done with a modified version of the ATEST module within AQ15c. This module requires, as input, an example to be tested and a set of class descriptions. The test example is extracted from the current problem. The meta-attributes which make up this meta-example are described in Table 4.1. The meta-rules used are either provided by a user, and/or learned in previous iterations

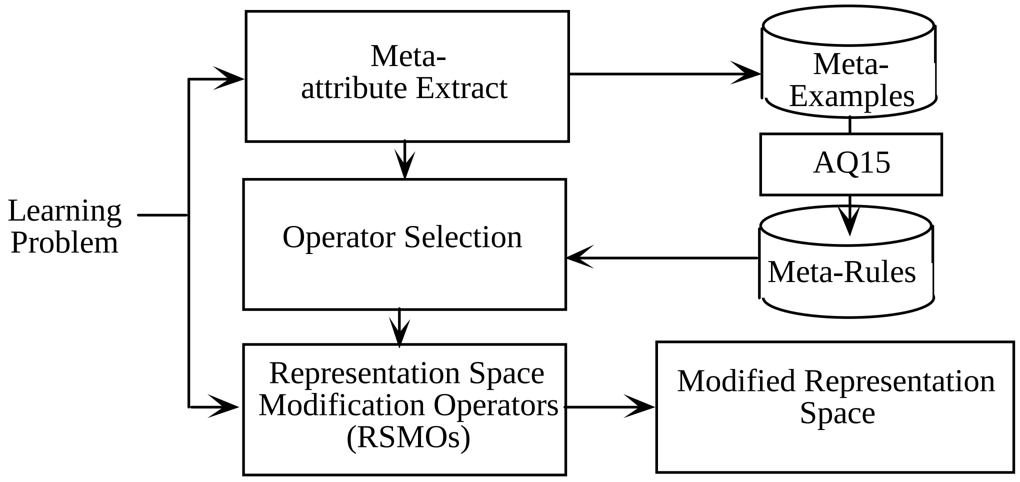


Figure 4.4 - Architecture of the Representation Space Modification Module

of the program and describe the conditions (properties of the learning problem) under which RSM operators have most improved the representation space.

4.5.1 Meta-attributes

Meta-rules are used to guide the selection of which representation space modification operator is appropriate for a given learning problem. Meta-rules relate the properties of the example dataset and the rule evaluation results on the secondary training set to the most appropriate operators. These rules can be initially provided by the user and incrementally improved by the system, or they can be learned by the system as it tries to solve the problems it faces. Meta-rules, and the meta-examples from which they are learned, are stated in terms of meta-attributes.

The meta-examples are described in terms of meta-attributes. Meta-attributes can be organized into four classes: 1) those characterizing types of the original attributes (numeric, multivalued nominal, Boolean, etc.), 2) those characterizing attribute quality, which is currently judged using an information gain ratio measure (Quinlan, 1993), 3) those characterizing the expected level of quality of the examples, and 4) those characterizing the changes in the performance of the

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	Yes, if data contains two or more multi-valued nominal attributes; No, otherwise
	Boolean_attributes_present	Yes, No	Yes, if data contains two or more Boolean attributes; No, otherwise
	Number_of_attributes	integer	Number of attributes in current space
Meta-attributes characterizing the attribute quality	Irrelevant_attributes_present	Yes, No	Yes, if data contains any irrelevant attributes; No, otherwise
Meta-attributes estimating ruleset performance	Last_simplicity	integer	Total number of selectors in learned rules
	Performance_estimation	Accuracy in percentag e (1..100%)	Predictive accuracy of the last ruleset generated from the primary training example set and tested on the secondary testing set.
	Selector-rule_ratio	real	total # selectors/ total # rules
	Average_number_of_uniquely _ covered_examples_per_rule	real	Sum of unique weights/ # of rules
	Example-rule_ratio	real	Total # of examples/ Total # of rules
	Average_number_of_internal_ disjunctions	real > 1.0	# values in selectors/ # of selectors
Meta-attributes estimating the quality of examples	Relative_rule_weight1	real <= 1.0	tweight rule #2/ tweight rule #1
	Relative_rule_weight2	real <= 1.0	tweight rule of last rule / tweight rule #1

	Training_examples_percent_of_total	real	# train given / # of examples possible

Table 4.1 - Meta-attributes for characterizing a learning problem

generated rules on the secondary training dataset. Table 4.1 presents a list of the current meta-attributes. All of these meta-attributes are extracted automatically by the system after initial hypotheses are learned.

4.5.1.1. Attribute Type

The applicability of some RSM operators depends on the type of the attributes. For example, arithmetic operators (for constructing new attributes from current attributes) applies to numeric attributes, while logical operators apply to Boolean and multi-valued nominal attributes. The type of attributes for which different RSM operators are available are currently all the types supported by the AQ learning module: linear (numeric integer, continuous), nominal (multi-valued and boolean) and structured.

4.5.1.2 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 ratio used in C4.5 (Quinlan, 1993), or by PROMISE (Baim, 1982).

The value of the meta-attribute "Irrelevant_Attributes_Present" is "Yes" if any one of the attributes in the given group of attributes has an information gain ratio greater than a user-defined minimum (default 0.1). This meta-attribute is useful for detecting irrelevant attributes, or if all attributes are below this threshold - for inferring that the given attributes may be correlated with each other, a situation in which attribute generation is useful.

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, 1994). In this measure an attribute is viewed as irrelevant if it 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). The AQ17-MCI method currently uses only the statistical information gain metric in order to judge attribute quality for use in the meta-attributes. However, the logical method based on hypotheses is used when the HCI method for feature selection is then invoked on the data.

There exist a number of alternative measures of individual attribute quality. One measure is *attribute utility* (Imam, et al., 1993). An attribute's 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. Another approach is the one used by rough set systems. In this method the quality of an attribute is measured by taking the difference in performance which occurs when rules are learned from the set including, vs. excluding, this attribute. This approach's strength lies in the fact that attribute quality is judged in the context of other attributes. This prevents problems which occur when an attribute is removed when it's average discriminatory power is low, but it is vital for a specific class, or when the quality of an attribute appears to be high even though it discriminates examples which are already easy to describe with other attributes (overlapping discrimination). These other approaches can be used in the AQ17-MCI framework, but currently are not available.

4.5.1.3 Example Quality

The quality of training examples is characterized in terms of three meta-attributes. The first two, relative-rule-weight1, and relative-rule-weight2, measure the distribution of examples in the space by looking at the relative coverage of the heaviest rule for a class against the coverage of the second heaviest rule and the lightest rule respectively. The focus of these meta-attributes it to try to capture the degree to which the training examples are distributed in a way that is easy to capture by the rules. If relative_rule_weight1 is high then that means the top rule covers many more examples than the second heaviest rule in the class. Assuming that training examples are provided in a proportion that reflects their true proportions, then this heavy cluster covers the most typical examples, while the other rules are covering less typical, and possibly noisy examples. The presence of these outlier examples, or noise, can mislead some attribute construction techniques, and statistically-based quantization. Thus, it may be wise to not apply these operators when this condition holds. If relative-rule-weight1 is low, then this means the examples are distributed fairly evenly across the space in a way that may be hard to capture. This is true for concepts which are spread across attributes such as 'm-of-n' or multiplexor problems. Such 'spread' concepts present a clear need for attribute construction methods which capture and explicitly characterize these interactions. Rule-weight2 has a similar motivation as rule-weight1 except that the conclusion that the concept is spread is even better supported when rule-weight2 is low.

The final example quality meta-attribute is 'Training_examples_percentage_of_total'. This meta-attribute is included to measure the confidence in the given training set as being prototypical of the real distribution. If this attribute is low, then the learning algorithm is forced to learn from a very sparse space. In such a case bias-strengthening (Utgoff, 1986; Gordon, 1990; Gordon and Desjardins, 1995) operations, such as attribute-removal or abstraction can be very dangerous. This measure also dramatically reveals one of the differences between synthetic examples and real-world domains. In the former, it is usually the case that a large percentage of the total learning space is labelled with class information, while in the latter this ratio is often very small.

4.5.1.4 Rule Performance

Rule performance is measured directly by determing the predictive accuracy in 'Performance_estimation' and the number of rules in 'Last_simplicity', and indirectly by rule coverage in 'Selector-rule-ratio', 'Average_number_of_uniquely_covered_examples' and 'Average_number_of_of_internal_disjunction'. The direct measurements are useful to guide the system into learning when conservative or dramatic changes are needed in the representation space. If the predictive accuracy is already high and the rules are fairly simple, then the concept representation space is fairly promising. In this case a conservative method such as constructing new attributes would be preferable to a more dangerous modification in which attributes are removed.

Selector-rule-ratio is used to help determine if attribute removal is possible. When AQ15c learns rules which are maximally general or of minimal length only the necessary attributes are included in the learned rules. If the selector-rule ratio shows that only a small percentage of the available attributes are being used in any given rule, then some attributes may be redundant or irrelevant. This signals the attribute-removal operators to be invoked.

The Average_number_of_uniquely_covered_examples_per_rule meta-attribute tries to capture the ease with which the rules covered the distribution of examples. If this number is high then the learned rules within a class are not overlapping - they are each covering a different clusters of examples. Attribute construction may be useful here to bring together these clusters of examples into fewer, larger and easier to describe clusters. If this meta-attribute is low, then rules are probably overlapping to a fairly high degree, but it is still hard to describe the cluster. Quantization may be used to here to reduce the distances between examples in the space by removing unnecessary detail. This may make rule coverage easier in the next generation. Example_rule_ratio provides similar information about the ability of the current hypotheses to cover the examples.

The average_number_of_internal disjunctions counts the number of values present, on average, in each selector of each learned rule. Based on the assumption that in a good representation space rules are simple, the presence of large numbers of internal disjunctions (e.g. [x1 = 1,3,5,7,9]) singnals the need for some kind of change. It may be that the attributes are measured with an excessive precision. To correct such a situation, the valueset of the attribute may be reduced, and the values of this attribute in the examples may be substituted by more abstract values. Overprecision can be reduced by abstraction. See Section 2.2.1.1 for a description of related work. AQ17-MCI uses a Chi-merge method to abstract attributes (Kerber, 1992).

4.5.2 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 meta-rules. This degree of match is calculated based on the relative number of conditions that match. Representation space modification operators are then ranked in decreasing order of meta-rule match. If no single meta-rule is the top rule, then selection between equivalent operators is based on the user's preference, if they are available, or random choice otherwise.

It may occur that the same RSM operator is repeatedly selected. In other words the search stagnates on a local maximum. AQ17-MCI attempts to prevent this by updating the database characterization after each ruleset evaluation. If this fails and the same operator is repeatedly selected, the next best matching operator is selected. 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.

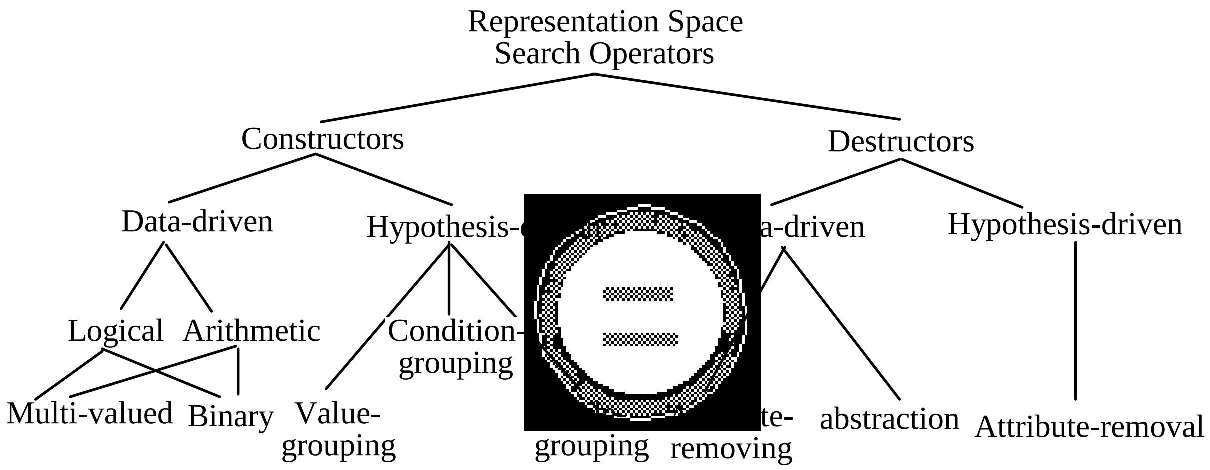


Figure 4. 5. A hierarchy showing the representation space modification operators in AQ17-MCI

Figure 4.5 presents a hierarchy of RSM operators used in AQ17-MCI. This hierarchy presents the types of modifications that may be performed within the attribute-value representation within which AQ operates. This hierarchical organization captures the relationships between RSM operators and allows selection rules to provide better guidance when confronted with new domains. The current MCI system has capabilities for both data-driven and hypothesis-driven construction of attributes, attribute removal by hypothesis, and statistical information and abstraction.

4.5.3 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 other systems (Bloedorn and Michalski, 1996; Wnek and Michalski, 1994) Some of these fundamental operators have been reported by others, notably Rendell and Seshu (Rendell and Seshu, 1990) The following MCI operators are equivalent to the terms used in Rendell: attribute removal (projection), and hypothesis-driven constructive induction (superpositioning).

4.5.3.1 Attribute Removal

Attribute removal makes a selection of a set X' of attributes from the original attribute set X. In MCI, attribute removal is done either in a hypothesis-driven approach using HCI-SEL (Wnek and Michalski, 1994), or in a data-driven approach using DCI-SEL (Bloedorn and Michalski, 1996). In the logic-based approach 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. In the data-driven approach an information gain ratio is used to calculate the 'quality' of an individual attribute. If the quality if less than a user-defined value (default 0.01) then the attribute is removed. An example of attribute removal is shown below.

Before:
Strong_box-events

color	material	age	shape
red	cardboard	12	square
blue	wood	3	rectangle
green	wood	5	square

After
Strong_box-events

material	age
cardboard	12
wood	3
wood	5

4.5.3.2 Abstraction

Abstraction is the merging of attribute values into intervals. Currently MCI implements only abstraction, based on the Chi-merge correlation between an attribute-value interval and the class. This method was proposed by Kerber (Kerber, 1992). Abstraction selects a set V' ℘ V (where V is the domain of A) of allowable values for attribute A. Abstraction can be used to reduce multi-valued large nominal domains, or real-valued continuous domains into domains consisting of only a small number of discrete values which represent intervals in the original representation. An example of abstraction on the attribute 'age' is shown below. In this example the ages 3..5 have been mapped to 0 and 12 has been mapped to 1.

Before:
Strong_box-events

color	material	age	shape
red	cardboard	12	square
blue	wood	3	rectangle
green	wood	5	square

After
Strong_box-events

color	material	age	shape
red	cardboard	1	square
blue	wood	0	rectangle
green	wood	0	square

4.5.3.3 Hypothesis-driven Attribute Generation

Hypothesis-driven attribute generation 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 they 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 and Michalski, 1991, Wnek and Michalski, 1994). Wnek and Michalski define three types 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, rule-groupings, and attribute removal. An example of hypothesis-driven attribute generation is shown below. In this example a new attribute 'ca1' has been generated. This attribute takes the value 1 when [material=wood][age=3..5] and 0 otherwise.

Before:

Strong_box-events

color	material	age	shape
red	cardboard	12	square
blue	wood	3	rectangle
green	wood	5	square

After

Strong_box-events

color	material	age	shape	ca1
red	cardboard	12	square	0
blue	wood	3	rectangle	1
green	wood	5	square	1

4.5.3.4 Data-driven Attribute Generation

Data-driven (DCI) methods build new attributes based on an analysis of the training data. One such method is AQ17-DCI (Bloedorn and Michalski, 1996). 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-driven attribute construction is performed by AQ17-DCI through maximum, minimum, average, and +, -, * and integer division.

The complexity of attribute DCI-Gen depends on the construction method used. The binary operators are O(A^2Ex) where A is the number of attributes and Ex is the number of examples. The functional operators are O(AEx) when knowledge about attribute units is used to determine the set of attributes. The #VarEQ(x) operator which actually builds many new attributes is

O(DAEx) where D is the size of the domain of all the attributes being combined. An example of attribute construction using the AQ17-DCI method is shown below. In this example a new attribute 'l*w' has been generated which is the product of length and width which is better understood as the area of one side of the box.

Before:
Strong_box-events

material	age	shape	length	width
cardboard	12	square	3	3
wood	3	rectangle	6	2
wood	5	square	4	4

After
Strong_box-events

material	age	shape	length	width	l*w
cardboard	12	square	3	3	9
wood	3	rectangle	6	2	12
wood	5	square	4	4	16

4.5.4 Storing Experience of Operator Selection

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 the RSM operator which made the useful modification of the representation space. If the quality remained constant or declined, then the representation space is returned to its previous state, and a new operator is selected. Thus learning does not currently occur when a rule fails to predict a successful operator. The failure of the first operator is only corrected implicitly when another operator is found to be successful for the current problem. If all RSM operators fail to improve the space, the vector characterizing the problem is stored as a positive example of the class representing no modifications.

AQ if:

[Last_simplicity = 3..15]

Interpretation:

Perform no change to the representation space if the learned rules are simple.

DCI-Quant if:

[Last_simplicity = 19..46] & [Training_examples_percent_of_total = 0]

Interpretation:

Perform quantization of the space if the last learned rules were moderately complex and the training examples provided is only a small percentage of the total possible space (common in real-world problems).

Figure 4.6. Examples of learned meta-rules

Given a set of classified meta-examples, new meta-rules can be learned or improved. Meta-rules are generated by AQ15c, or provided by the user. Learning of meta-rules is invoked at the conclusion of a learning session. This occurs when a problem is solved (the learned rules exceed the user thresholds for quality), or no more operators can be tried. 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 Figure 4.6. These meta-rules are learned from a set of fourteen meta-training examples representing fourteen datasets. These datasets and the predictive accuracy of these learned meta-rules are described in Chapter 5.

4.6 Summary

This chapter described in detail the AQ17-MCI system which implements multistrategy constructive induction. In this system, learned hypotheses are repeatedly evaluated on predictive accuracy and complexity after various modifications to the representation space are made. Those modifications that result in improved hypotheses are retained, while those that do not are discarded - a greedy search with lookahead of one. Different operator control methods can be used to select RSMO's including: a) random selection when no knowledge, or meta-rules is provided, b) fixed ordering (e.g. First construct new attributes via DCI-Generate and HCI-Generate, select relevant attributes with HCI-Select, then abstract attribute values with DCI-Quant, and finally select attributes with DCI-Select) and c) learned meta-rules. Chapter 5 describes experiments applying AQ17-MCI on a variety of synthetic and real-world problems.

CHAPTER 5 EXPERIMENTS

5.1 Experimental Goals

This chapter provides empirical support for multistrategy constructive induction as described in this thesis. The goal of this chapter is to show that multistrategy constructive induction a) outperforms traditional selective induction alone, and b) outperforms any single-strategy constructive induction method from the set that make up MCI. The performance metric used here is *predictive classification accuracy*. Predictive classificicication accuracy is measured as the percentage of correct classifications made by the final hypothesis for a set of test examples. Test examples are randomly selected and were unavailable during hypothesis generation. The performance of MCI will be evaluated in two ways: 1) its ability to solve a wide variety of representation space problems within one application framework, and 2) its ability to solve problems with multiple pathologies requiring the individual representation space modification operators to work together.

5.2 Experimental Design

The single most important factor influencing the success of inductive learning is the quality of the representation space (RS). Here the representation space is defined as being the space of descriptors and their values. Although also important, the qualities of the examples themselves, such as their typicality and accuracy, are not directly a part of the representation space, and as such are not addressed by the *representation space* modification operators discussed here. The qualities of the RS being focused on are:

a) **Suitability of attribute-values to classification task.** If attributes are measured with too little precision then concept boundaries are blurred and difficult to discriminate. Conversely too much precision can artificially expand the distances between examples increasing the possibility that important patterns will be missed (i.e. failure to see the forest for the trees). Overprecision is most often seen in problems with many numeric attributes such as those present in problems in computer vision or economics.

b) **Relevance of the attributes to the classification task**. The true correlation between the class attribute and the other attributes is often difficult to determine without extensive data or background knowledge. Having only truly relevant attributes available in the RS increases the possibility that the learned hypotheses will truly capture strong patterns and be predictively accurate. However, often even experts disagree, or don't know what factors are relevant to a classification task. For example, in segmenting an image into classes of objects, or retrieving relevant documents from a large text collection, attribute relevance can be a serious problem.

c) **Independence of attributes**. Selective induction learning algorithms assume that all the attributes are independent. Such algorithms cannot express even very simple correlations between attributes such as 'height = width'. The independence assumptions is often violated for real-world domains because every attribute which is thought to have an effect on the outcome is included regardless of their likely non-independence.

Learning in a representation space that satisfies these criteria is easy. Unfortunately, designing a

RS for a given problem which satisfies these criteria is often extremely difficult, especially for real-world problems. In fact in many real-world applications, knowing the relevant attributes, and whether they are independent or not is part of the problem.

In order to expand the applicability of machine learning to these real-world problems, an automated, or partially automated method for finding an improved representation space must be found. The MCI approach has been designed to perform this automated search for an improved representation space and to correct the problems that arise when each of the previous criteria are not satisfied.

The following experiments are designed to evaluate the ability of the representation space modification operators in MCI to find an improved representation space. In the first set of artificial problems the necessary transformations are known. This allows the experimental results to be compared with predicted outcomes. In the latter examples, the exact form of the goal concept is not known. For these problems the results of MCI operations are evaluated based on the change in predictive accuracy, and the knowledge generated (e.g. the meaning of the attributes generated)

Evaluating the MCI method, when it is uses learned meta-rules to guide the search for an improved representation space, is complicated by the fact that the performance is determined by the quality of the knowledge available to it in the form of meta-rules. This knowledge cannot be obtained without some input from a user either in the construction of the meta-rules directly or indirectly from the selection of meta-examples. In either case, it can be argued that the true performance of the overall system cannot be accurately determined because of the bias provided by outside knowledge. Although it is impossible to obtain an “average” measure of performance because of this user bias, it is possible to obtain an upper performance bound. This can be achieved by evaluating all the available representation space modification (RSM) operators for a given problem. The best individual performance is then selected as the performance of MCI. If the meta-rules are perfectly accurate in predicting the correct class than MCI will perform at this upper level. The quality of the meta-attributes in extracting relevant cues for operator selection, and in the meta-rules for predicting operator class is separately evaluated in section 5.5.

5.3 Synthetic Problems

This set of artificial problems was generated to carefully evaluate the effectiveness of MCI to overcome a learning problem made difficult by: a) overprecise measurement of attribute values, b) the presence of irrelevant attributes, and 3) the presence of non-independent attributes. If the necessary transformation are made, the simple DNF rule which is the design goal concept, should easily be found by the underlying induction algorithm AQ15c.

5.3.1 Description

This problem set consists of four 2 term DNF functions (e.g. the goal concept consists of 2 disjuncts) first described in (Bloedorn, Michalski and Wnek, 1994) The goal concept for each of the six problems is the same. However, in all but the first case the goal concept has been obscured by a different type of problem. The three different problems, corresponding to the three criteria of section 5.2 are: 1) Overly large attribute domain sizes in which like-labelled examples are distant in the representation space (criteria a) 2) Irrelevant attributes (criteria b) and 3) Dependent attributes (criteria c). A more detailed description of each of the 4 problems (the original and each of the three described) is given below. The attributes used in this problem set have between two and 60 different values. The values are linearly ordered. The domains of attributes x1, x2, and x3 have domains of [0..5] in the base case while x4 and x5 have a domain of [0..1]. The goal of this set of experiments is to determine the effectiveness of the available set of RSM operators to overcome the different difficulties introduced to the simple base problem.

1) Problem t0 original DNF

Positive class: [x1=3,4] [x2=1..3][x3=1,2] v [x3=3,4][x4=1][x5=1]

2) Problem t1 (Overprecision: the domains of attributes x4 and x5 have been expanded 10-fold)

Positive class: [x1=3,4] [x2=1..3][x3=1,2] v [x3=3,4][x4=10..19][x5=10..19]

3) problem t2 (inappropriate attributes: the decimal value of x3 has been mapped using a 4 place parity coding, e.g. 3 = 0 1 1 1. The selection of a particular equivalent coding is random)

Positive class: [x1= 3,4] [x2=1..3][[#attributes(x6..x9)=1]=1,2] v
[[#attributes(x10..x13)=1]=3,4][x4=1][x5=1]

4) Problem t3 (the first 30 attributes are irrelevant)

Positive class: [x31=3,4] [x32=1..3][x33=1,2] v [x33=3,4][x34=1][x35=1]

5.3.2 Method

All five of the RSM operators, DCI-Gen, DCI-Quant, DCI-Sel, HCI-Gen and HCI-Sel were applied to problems t0, t1, t2 and t3. Each problem was evaluated using 10-fold cross validation (Weiss and Kulikowski, 1991) Significance is calculated using a two-tailed student t-test.

5.3.3 Results

Table 5.1 shows the predictive accuracy of AQ alone (no representation space modification) and AQ17-MCI for the four DNF problems. By combining the performance of multiple RSM operators, AQ17-MCI performs better than AQ alone on all the corrupted learning problems.

Additionally, Table 5.2 shows that no single RSM operator can correct all three learning problems, showing that some combination of strategies is required to achieve high performance over the range of problems.

Table 5.1 - Results of AQ and AQ17-MCI on 3 artificial problems

Problem Method	Baseline DNF	expanded domain	distributed coding	irrelevant attributes
AQ				
Avg. Accuracy	100.0	96.8	93.9	87.9
Avg. #Rules	8.1	15.6	33.5	18.6
Avg. #Selectors	17.3	45.6	146.0	172.3
Avg. L. Time (sec)	0.5	13.9	3.82	440.2
AQ17-MCI	**AQ only**	**Quant**	**Generate**	**Select**
Avg. Accuracy	100.0	100.0[1]	98.6[3]	99.6[2]
Avg. #Rules	8.1	8.9[1]	14.2[1]	20.5
Avg. #Selectors	17.3	20.4[1]	51.3[1]	135.4
Avg. L. Time (msec)	0.5	3.31[1]	8.5[1]	121.1[1]
Avg. CI Time (sec)	0	0.2	6.0	0.9

([1]: signifigance $\alpha = 0.01$ [2]: signifigance $\alpha = 0.05$ [3]: signifigance $\alpha = 0.1$)

Problem T0 (Original DNF)

The base learning problem (t0) is clearly easy to learn. The rules learned without modification to the representation space average 8.10 rules with 17.3 selectors, take 0.45 milliseconds to learn and have 100% predictive accuracy. The learned rules for the positive class exactly match the goal target concept given above.

Problem T1 (Expanded Domains)

In problem t1 the simple DNF problem of t0 has been complicated by artificially expanding the size of the attribute domains of x4 and x5 10-fold. In the original representation of the problem these attributes were binary, but now they each take 10 values. As these attributes are only used in one of the disjuncts in the goal concept it would be expected that learned rules would be more complex, but have about the same predictive accuracy as in t0. This is exactly what occurs: AQ generates an average of 15.6 rules (92% increase over t0), 45.6 selectors (163% increase) in 13.91 milliseconds with a predictive accuracy of 96.8%. However, the DCI-QUANT system was able, to a high degree, correctly identify which intervals were meaningful and make the necessary repair to the representation. The rules 'fixed' by this transformation were 100% accurate (± 0.0), had 8.9 rules, with an average of 20.4 selectors. These rules are only slightly more complex than the rules learned in the original space and required a learning time of 3.31 milliseconds.

Problem T2 (Attribute Interaction)

In problem t2, t0 has been complicated by introducing non-independent attributes. The simple description of the goal concept has been 'blurred' (Rendell and Ragavan, 1993) by spreading the values of x2 and x3 across 8 additional attributes (x6..x9 now represent the value of x2; and x10..x13 now hold the value of x3). This blurring has reduced the predictive accuracy to 93.9%, increased the number of rules from 8.10 to 33.5 (313% increase), increased the number of selectors from 17.3 to 146 (743% increase) and increased the learning time from 0.45 milliseconds to 3.82 msec (748% increase). However, this problem has been corrected by the

DCI-GENERATE method using the SUM operator. SUM uses information about attribute units to guide construction of new attributes. With new attributes, DCI-Generate was able to learn rules which were 98.6% accurate (±1.82 $\alpha = 0.1$), had 14.2 rules and 51.3 selectors.

Problem T3 (Irrelevant Attributes)

In this problem t0 has been complicated by 30 additional, randomly generated attributes. The learning algorithm must select the important attributes from the irrelevant. AQ15c performs attribute selection well, but with large amounts of irrelevant attributes, even AQ15c's performance can degrade. This is what was found with problem t3: AQ learned rules that were only 87.9% predictively accurate (± 6.16 α=0.1), had 18.6 rules, 172.3 selectors and took 440.15 milliseconds to learn. DCI-Select was able to perform the correct transformation to the representation space resulting in rules that were significantly more accurate (99.6% ± 0.87 α=0.01), had shorter rules, 135.4 selectors and took significantly shorter time to learn 121.06 milliseconds (± 18.71).

In addition to these results, the effect of other MCI operators on these four datasets was measured. These results (Table 5.2) show that although some transformations are beneficial, some are not. For example, attribute generation can be incorrectly encouraged, by expanded domains and irrelevant attributes, into generating attributes that appear to be relevant, but really are not. Attribute abstraction can also be harmful to learning if applied to problems which are already well suited as shown in the results for DCI-Quant in the Baseline DNF case. This would not occur in AQ17-MCI, because it checks to see if modification to the representation space is needed by learning hypotheses in the initial space. This kind of extra processing and damamging effects are avoided by this simple initial check. The results for DCI-Select also point out the difficulty of detecting attribute relevance in the presence of attribute interaction. In this case the rules learned in the DCI-Select transformed space were significantly worse than the rules learned in the initial space (predictive accuracy: 93.9 vs 69.0)

Table 5.2 - Performance of individual RSM operators on 3 DNF problems

Problem Method	Baseline DNF	expanded domain	distributed coding	irrelevant attributes
DCI-Generate				
Avg. Accuracy	100.0	92.0	98.6	85.2
Avg. #Rules	7.8	16.7	14.2	17.7
Avg. #Selectors	18.4	65.2[2]	51.3	191.3
Avg. L. Time (sec)	1.0	61.9[1]	8.5	836.7[1]
Avg. CI Time (sec)	.02	1.6	6.0	4.9
DCI-Quant				
Avg Accuracy	85.9[1]	100.0	94.7	87.0
Avg. #Rules	8	8.9	33.5	29.3[1]
Avg. #Selectors	17.9[3]	20.4	149.1	223.6[3]
Avg. L. Time (sec)	0.3[1]	3.31	5.5[1]	233.8[1]
Avg. CI Time (sec)	6.7	0.2	0.2	3.3
DCI-Select				
Avg. Accuracy	91.3[1]	89.2[2]	69.0[1]	99.6
Avg. #Rules	10.4	19.7[1]	2.0[1]	20.5
Avg. #Selectors	29.0[2]	57.2[2]	4.0[1]	135.4
Avg. L. Time (msec)	0.4[1]	14.2	0.2[1]	121.1
Avg. CI Time (sec)		0.1	0.2	0.9
HCI-Select				
Avg. Accuracy	100.0	96.8	93.9	89.7
Avg. #Rules	8.1	15.6	33.5	20.1
Avg. #Selectors	17.3	45.6	146.0	191.4
Avg. L. Time (msec)	0.5	13.9	3.82	631.8[1]
Avg. CI Time (sec)	3.7	3.7	4.9	5.2
HCI-Generate				
Avg. Accuracy	100.0	95.8	94.4	93.1
Avg. #Rules	7.5	7.5[1]	13[1]	7.2[1]
Avg. #Selectors	15.8	19.6[1]	46.9[1]	29.8[1]
Avg. L. Time (msec)	0.6	6.3[1]	3.11[1]	116.24[1]
Avg. CI Time (sec)	3.7	4.3	5.2	9.1

([1]: signifigance $\alpha = 0.01$ [2]: signifigance $\alpha = 0.05$ [3]: signifigance $\alpha = 0.1$)

5.4 Real Problems

This section provides examples of the performance of AQ17-MCI on two real-world domains. These domains are segmentation of visual scenes, and GNP change prediction from economic and demographic data. These domains show how AQ17-MCI produces the best results when multiple RSM operators are used together. This supports somewhat the claim that the problems identified (overprecision, irrelevant attributes and attribute-interdependencies) are present in real problems.

5.4.2 World Economics

5.4.2.1 Description

The ability to detect economic trends is an important and difficult problem. Like many real problems finding the right representation is very difficult. In the experiments described here the goal is to find patterns in demographic and economic data which can be used to predict GNP (Gross National Product).

Table 5.3 - Initial representation Space of GNP Problem

Population (age 10-14) as % of total
Population *age 15-64) as % of total
Urban Population growth rate
Urban/rural growth difference
Crude birth date (per thousand population)
Crude death rate (per thousand population)
Agricultural land as % of totall land area
Net deforestation rate (annual %)

Food producation per capita (1979-1981 = 100)
Energy consumption per capita (kg of oil equivalent)

Attributes characterizing a country in a given year. Training examples provide a characterization of a country for 5 years for a total of (11*5 = 55 attributes)

As a first step toward predicting GNP for a given country and year, the following problem was formulated in which the goal was to characterize levels of GNP change based on economic and demographic attributes. The data were obtained from a World Bank database (Bloedorn and Kaufman, 1996). Although the entire database contains economic and demographic records for the countries of the world from 1965 to 1990, (Kaufman, 1994) these experiments focus on a smaller set of countries during the period from 1986 to 1990.

5.4.2.2 Method

In the experiment we considered 41 countries based on their geographic distribution (x from the Americas, x from Europe, etc.) and the completeness of their records in the database. Changes of GNP were quantized into four equal-intervals: low (0 to 0.5625), medium (0.5626 to 1.125), high (1.126 to 1.6875) and very high (over 1.6875). The countries were described by 11 attributes, each sampled over a period of 5 years. Thus each country was described by 55 attributes. The initial representation space for this problem is shown in Figure 5.3. The quality of the generated rules was evaluated using a 10-fold cross-validation method (Weiss and Kulikowski, 1991).

Selective Induction

The standard approach to solving this problem is to apply a selective induction learning algorithm to the raw data directly. The results obtained for this approach are shown below. This is the baseline performance.

Table 5.4 - GNP results after learning in the original representation space

	Avg. Accuracy (%)	Avg. # Rules	Avg. # Selectors	Avg. Learning Time (sec)
AQ alone	46.2	6.72	19.5	171.9

Single Strategy Constructive Induction

All available RSM operators were applied to the problem with the results shown in Table 5.5. The HCI methods have a built in check for improvements in predictive accuracy over the baseline case. For this problem the transformations introduced by both HCI-Gen and HCI-Sel were no better than the baseline. For the data-driven methods, the transformations did make a difference. DCI-Gen, on average, added 14 new attributes for each problem. Examples of the attributes generated are shown in table 5.6. DCI-Quant reduced the domain size from an average of 15.4 to and average of 5.2. This not only caused a reduction in learning time from 171.9 seconds to 11.2 seconds, but an increase in average predictive accuracy from 46.2% to 58.0%. DCI-Sel removed an average of 7 attributes from each problem, but this resulted in little overall improvement in learned hypotheses. DCI-Quant also effectively removed 21 attributes from the space by reducing their domains size to a single value. This likely contributed to the significant reduction in learning time.

Table 5.5 - GNP results after applying individual RSM operators to the original problem

	Avg. Accuracy (%)	Avg. # Rules	Avg. # Selectors	Avg. Learning Time (s)	Avg. CI Time (s)
DCI-Gen	64.3	5.9	18.3	205.5	0.3
DCI-Quant	58.0	7.2	20.1	11.2	1.3
DCI-Sel	46.2	6.7	19.5	122.7	0.2
HCI-Sel	46.2	6.7	19.5	171.9	2.0

HCI-Gen	46.2	6.7	19.5	171.9	1.6

(^{1}: signifigance $\alpha = 0.01$ ^{2}: signifigance $\alpha = 0.05$ ^{3}: signifigance $\alpha = 0.1$)

Multistrategy Constructive Induction

Table 5.5 shows examples of applying multiple RSM operators to the same problem. The best results occurred with the quantization of attributes followed by the generation of new attributes. These results are interesting for two reasons. The first reason is that the search for an improved representation space is shown to not be best searched by a simple greedy method; although the DCI-Gen space seemed best at first (Table 5.4), the DCI-Quant space followed by DCI-Gen (with DCI-Gen adding an average of 15.6 attributes) actually resulted in higher predictive accuracy and much shorter learning time. Good meta-rules may be able to overcome this problem by forcing the system to perform DCI-Quant first based on the characteristics of the problem. The other point is that these results also show that the order of RSM operators is important. This further supports the need for control methods which are not fixed in order. Additional experiments involving combinations of three RSM operators, but these did not result in an improvement.

Table 5.6 - GNP Results after applying multiple RSM operators

Method	Avg. Accuracy (%)	Avg. # Rules	Avg. # Selectors	Avg. L. Time (s)	Avg. CI Time (s)
DCI-Gen-> DCI-Sel	64.3	5.9	18.3	188.8	0.3
DCI-Gen->DCI-Quant	43.7	7.0	19.6	77.9	2.1

DCI-Sel-> DCI-Gen	66.1	5.9	18.4	187.5	0.5
DCI-Sel-> DCI-Quant	53.4	7.5	21.2	38.6	1.5
DCI-Quant-> DCI-Gen	76.3	7.0	17.9	28.22	0.7
DCI-Quant-> HCI-Gen	58.0[3]	7.2	20.1	9.76	3.2

([1]: signifigance $\alpha = 0.01$ [2]: signifigance $\alpha = 0.05$ [3]: signifigance $\alpha = 0.1$)

5.4.2.3 Results

The results for DCI-Quant followed by DCI-Gen (Table 5.6) shown an approximately 80% increase in predictive accuracy, slightly less complex rules, and significantly lower learning time over rules learned in the original space (Table 5.3). The multiple transformations were not only very useful for prediction, but were also easy to understand. Examples of generated attributes are shown in table 5.8. These new attributes are natural combinations of the originals based on operators like Average and Minus. An example of a learned rule in the abstracted space

Table 5.7 - Summary of results for GNP problem

Method	Avg. Accuracy (%)	Avg. # Rules	Avg. # Selectors	Avg. L. Time (s)	Avg. CI Time (s)
AQ only	46.2	6.7	19.5	171.9	0
DCI-Gen (Best Single RSM)	64.3	5.9	18.3	205.5	0.3
DCI-Quant-> DCI-Gen (Best Multiple RSM)	76.3	7.0	17.9	28.22	0.7

with new attributes is shown below. This result shows that the operators in AQ17-MCI can be used together in order to produce a final result that no single method could.

Countries with very high increase in GNP:

[Death Rate is low] &

[*AVG(%PopulationAgeBracketB)* [vii] is very high] &

[*AVG(Population Growth Rate)* is low] OR

[*AVG(Urban Population Growth)* is very low] &

[*AVG(Urban vs. RuralGrowth Difference)* is very low]

Interpretation: Countries with a very high increase in GNP are characterized by low death rate, the average perecentage of the population age 15 to 64 years is very high and the overal popoulation growth rate is low, or the average urban population growth is very low, and the average difference between urban and rural population growth rate is very low.

Table 5.8 - Examples of new relevant attributes constructed by DCI-Generate

Name	**Operator used**	**Description**
Avg (Population growth rate)	Average	The average population growth rate 1986 to 1990
ChgeEnergyCons86-88	Minus	Change in energy consumption of a country between 1988 and 1986
Avg(%Population Age Bracket B)	Average	The average percentage of the population age 15 to 64 years old
AveEnergyCons86-90	Average	Average Energy Consumption of a country between 1986 and 1990

5.4.3 Computer Vision

[vii]Generated attributes are shown in italics

5.4.3.1 Description

This section details an application of multistrategy constructive induction to the interpretation of natural scenes. In this problem the goal is develop a method which can accurately distinguish objects in outdoor scenes under varying perceptual conditions. The approach used here is to learn characterizations of classes of natural objects (sky, trees, road) from images that have been labelled. These characterizations, based on features extracted for pixel windows, can then be applied to new scenes in order to predict the presence of natural objects.

5.4.3.2 Method

In the experiment the input to the learner was a training image which includes selected examples of the visual concepts to be learned: sky, trees and road. A windowing operator, of size 5x5 scanned over the training area, was used to extract a number of attributes including: color intensity (red, green and blue), horizontal and vertical line, high frequency spot, horizontal and vertical v-shape, and Laplacian operators. The quality of the generated rules was evaluated using a 10-fold cross-validation method. This data set has 450 examples equally distributed between the three classes.

Selective Induction

The standard approach to solving this problem is to apply a selective induction learning algorithm to the raw data directly. The results obtained for this approach are shown in Table 5.9. This is the baseline performance.

Table 5.9 - Computer vision results after learning in the original representation space

	Avg. Accuracy (%)	Avg. # Rules	Avg. # Selectors	Avg. Learning Time (sec)
AQ alone	72.5	27.7	94.8	231.7

Single Strategy Constructive Induction

An exhaustive algorithm was used to find the best transformations to the representation space. In phase 1, all the available RSM operators were applied to the raw data (Table 5.10). This table shows a dramatic improvement resulting from quantization of the data. The DCI-Quant operator reduced average attribute domain size from 256 to 14. The rules learned after the attribute values had been quantized were significantly more predictively accurate, and the learning time was significantly shorter than for rules learned in the original space. However, rule complexity increased from an average of 27.7 rules to 34.6 rules and there was a significant increase in number of selectors used in the rules.

The rules learned from the space expanded by DCI-Generate were also significantly more accurate than the rules learned in the original space. DCI-Generate added on average 10 new attributes, the strongest of which described absolute and relative differences in the amount of red, green and blue color intensities. Given the green trees, the dark road and the blue sky present in the training images this is not surprising. The tree class included new attributes that stated: [green > red] and [green > blue]. The introduction of these new attributes to the representation space resulted in a significant improvement to all aspects of the resulting rules. In the new representation space significantly more predictively accurate rules, fewer in number, and of less complexity were learned in shorter time than in the original representation space.

Table 5.10 - Computer vision results after applying individual RSM operators

	Avg. Accuracy (%)	Avg. # Rules	Avg. # Selectors	Avg. L. Time (s)	Avg. CI Time (s)
DCI-Sel	75.3	34.7	74.6[1]	215.1	3.1
DCI-Quant	85.9[1]	34.6	114.7	10.8[1]	5.7
DCI-Gen	87.1[1]	18.5[1]	63.5[1]	171.1[1]	8.1
HCI-Gen	71.4	24.9	89.1	359.0	5.9
HCI-Sel	72.5	27.7	94.8	280.6	5.4

([1]: signifigance $\alpha = 0.01$ [2]: signifigance $\alpha = 0.05$ [3]: signifigance $\alpha = 0.1$)

The transformations the representation space made by the other operators were not as useful as those made by DCI-Quant and DCI-Gen. DCI-Sel was the next most useful transformation. DCI-Sel consistently removed attributes x4..x8, which are attributes like: horizontal and vertical line, high frequency spot, horizontal and vertical v-shape that describe the patterns within the 5x5 extraction window. This resulted in a slight increase in predictive accuracy, a significant reduction in the number of selectors present in the rules and in learning time, but an increase in the number of rules generated. HCI-Sel made only small changes in the representation space of the problem removing only two attributes over the course of the ten runs. HCI-Gen was also conservative in its modifications constructed new attributes on only three of the ten runs. Two of these three resulted in decreases in predictive accuracy while the third resulted in no change.

These single-strategy results are encouraging and lead to the investigation of combinations of operators, especially attribute generation through DCI-Gen combined with abstraction of the space through DCI-Quant. The next section describes these experiments.

Multistrategy Constructive Induction

Table 5.11 shows the results from combinations of the three RSM operators: DCI-Gen, DCI-Quant, and DCI-Sel that made significant improvements to the representation space. The previous section showed how initial abstraction of the space was useful, but may have been suboptimal given the increase in rule complexity. This finding was reinforced when DCI-Gen was run on the abstracted space. Although the new attributes resulted in fewer rules, they were less accurate than those learned in the DCI-Quant only space, and were more complex. The contraction of the space may be removing some important information. If the operators are reversed and DCI-Quant is applied to the space already expanded by DCI-Gen, the space is significantly improved in almost all respects: Both learning time and predictive accuracy are now significantly better than in both the original space *and* the DCI-Gen only space, while rule complexity and number slightly increased from the DCI-Gen only space, but is still significantly smaller than in the original representation.

DCI-Sel used after DCI-Gen also results some improvements. DCI-Sel after DCI-Gen results in a small increase in predictive accuracy over DCI-Gen alone, a significant decrease in learning time and the number of selectors used, but an increase in the number of rules. In this space as in the original representation, DCI-Sel removed attributes x4..x8. DCI-Gen followed by DCI-Quant and then DCI-Sel was tried, but resulted in no improvement in predictive accuracy of the learned rules.

Table 5.11 - Computer vision results after applying multiple RSM operators

	Avg. Accuracy (%)	Avg. # Rules	Avg. # Selectors	Avg. L. Time (s)	Avg. CI Time (s)
DCI-Quant ->DCI-Gen	85.4[1]	25.6	147.8	283.0	5.7+1.4 = 7.1
DCI-Gen ->DCI-Quant	93.4[1(3)]	20.5[1]	63.7[1]	104.5[1(1)]	8.1+19.3=27.4
DCI-Gen->DCI-Sel	90.3[1]	25.7	53.6[1(3)]	142.2[1(2)]	5.7+1.0 = 6.7

([1]: signifigance $\alpha = 0.01$ [2]: signifigance $\alpha = 0.05$ [3]: signifigance $\alpha = 0.1$)

5.4.3.3 Results

The multiple transformations generated by AQ17-MCI, attribute generation followed by attribute quantization, resulted in rules which are significantly more accurate, can be learned much faster, and are much less complex than rules learned in the original representation.

The DCI-Gen combined with DCI-Quant result suggests that there is both an interaction between attributes and an excess of detail in the original representation. By performing DCI-Gen first, this interaction appears to be at least partially captured, and the abstraction operator of DCI-Quant can now safely perform its operation without looking at the context provided by the other attributes. Because DCI-Quant (using the Chi-merge algorithm) views each attribute independently it may remove information that is important to classification. Doing DCI-Gen first, this danger is reduced. Before any general conclusions can be drawn about the best ordering of RSM operators it must be remembered that the previous GNP problem was best solved using a DCI-Quant, DCI-Gen ordering. The conclusion that can be drawn from this is simply that some patterns are best described in the original formulation of the problem, and some only become apparent after abstraction. If abstraction is sensitive to interactions betweem attributes it may be possible to eliminate such ordering effects, but such a method would have to search an enormous space of both combinations and abstraction levels. An approach which more tightly couples the search for combinations and abstraction level is an interesting and important area for future research. In the meantime it reinforces the need to flexibly combine RSM operators and not hard-code certain orderings. The learned meta-rule approach of AQ17-MCI has such a flexible capability.

The attributes constructed by DCI-Gen were not only useful for classification, but also have an easily interpretable mearning as differences in color intensities. The difference in color intensity between red and green, and between green and blue, were consistently in the top three most informative attributes as measured by information gain Table 5.12. The difference between red and blue was also generated, but was not found to be of high discriminatory power.

Table 5.12 - Examples of new relevant attributes constructed by DCI-Generate

Name	**Operator used**	**Description**
red - green	subtraction	intensity difference between red and green
green - blue	subtraction	intensity difference between green and blue

Multistrategy constructive induction helped not only to increase the prediction accuracy, but also generated a number of meaningful new attributes. These transformations explicitly found one combination of color intensities based on difference that was useful. This ability to clearly see and understand the transformations made by the operators allows researchers to better understand the results. It may also inspire the use of other representations. In this case a hue, intensity and saturation representation may be used as well for example. The results from applying AQ17-MCI to this computer vision problem shows that the operators in AQ17-MCI can be used together in order to produce a final result that no single method could.

5.5 Learning Meta Control Rules

This section describes the measures taken to acquire and evaluate meta-rules used to control the selection of representation space modification operators. These meta-rules describe the conditions under which each of the operators produces the greatest improvement to the representation space. These conditons are built up from meta-attributes extracted for both the data and initial hypotheses learned from the given problem.

The quality of the meta-rules is measured by their ability to predict the correct operator ‘class’ for a given vector description of a learning problem. This ability is primarily determined the quality of the meta-representation space although different learning algorithms may perform differently on the same data. A complete search of the learning bias space and the meta-attribute space is outside the scope of this thesis. In this thesis, the meta-learning algorithm will be fixed to AQ15c, and the bias will be set to the default criteria.

5.5.1 Description

The meta-attributes used to describe the given learning problem at hand are given in Table 4.1. The values for the meta-attributes are extracted for 11 artificial and 3 real datasets (Table 5.13).

These problems were selected because they represented examples of both a) when each of the five available representation space modification operators perfomed best and b) when no modification

Table 5.13 - Descriptions of meta-examples used in evaluating meta-attributes

Key Name	**Description**	**Best RSM operator**	**Source**
M1	Monk Problem #1	DCI-GENERATE	Thrun et al., 1991
M2	Monk Problem #2	DCI-GENERATE	Thrun et al., 1991
NIM2	Noisy and Irrelevant Monk Problem #2	DCI-SELECT	Bloedorn et al, 1993
t1.10	Irrelevant DNF t1 with 10% training	none - AQ	Bloedorn, 1996
t1.20	Irrelevant DNF t1 with 20% training	DCI-SELECT	Bloedorn, 1996
t1.60	Irrelevant DNF t1 with 60% training	DCI-SELECT	Bloedorn, 1996
t3.60	Overprecise DNF t3 with 60% training	DCI-QUANT	Bloedorn, 1996
t3.40	Overprecise DNF with 40% training	DCI-QUANT	Bloedorn, 1996
Security	Unix user profiles	DCI-QUANT	Maloof and Michalski, 1995
Hepatitis	Hepatitis prediction	DCI-GENERATE	UC-Irvine
t1.40	Irrelevant DNF t1 with 20% training	HCI-SELECT	Bloedorn, 1996
M1DCI	Monk Problem #1 with DCI generated (x1=x2) attribute	none-AQ	Bloedorn, 1993

M2DCI	Monk Problem #2 with DCI generated #VarEQ(1) attribute	none-AQ	Bloedorn et al, 1993
Scale_hepatitis	Hepatitis problem after QUANT abstraction	HCI-GENERATE	Bloedorn, 1996

was best. The predictive accuracy of the learned meta-rules was evaluated using leave-one-out.

5.5.2 Meta-Rule Evaluation: Holdout

This section describes how the meta-rules learned from the meta-example dataset of Table 5.13 were evaluated. Two tests were performed. In both tests the training data consisted of the entire set of examples less one. This single held-out example was used for testing. Evaluation of the correct class was based on the degree of match between the test example and the learned meta-rules. If the correct class had the highest degree of match the accuracy was 100%, otherwise the accuracy was 0. In this first test, all 6 distinct classes of representation space modification were used. The six classes are: —

1. None - No change needed, AQ alone performed well
2. DCI-GENERATE- Data-driven attribute construction
3. DCI-QUANT- Data-driven attribute value abstraction
4. DCI-SELECT- Data-driven attribute selection
5. HCI-GENERATE- Hypothesis-driven attribute construction
6. HCI-SELECT- Hypothesis-driven attribute selection

Because there are many continuously valued meta-attributes it became necessary to abstract these many values into more meaningful ranges. This process was done using DCI-Quant. The intervals found to be important, for the numeric-valued meta-attributes, are shown in Table 5.14.

Rules were learned over 14 problem sets using a leave-one-out method. The meta-rules learned for one of these sets, using the discretized domains is shown below (the only example of the HCI-Select class was held out, so there is no learned rule for that class).

Table 5.14 - Intervals found for meta-attributes using DCI-Quant

Attribute Name	Intervals	Attribute Name	Intervals
Average_number_of_internal_ disjunctions	[1.0..1.2] [1.4..3.8]	Number_of_attributes	[5..28] [55]
Relative_rule_weight1	0.13..0.49] [0.62..0.99]	Selector_rule_ratio	[1.0..1.3] [1.92..4.31]
Relative_rule_weight2	[0.03..0.16] [0.18..0.34]	Average_number_of_ uniquely_covered_exam ples per_rule	[3.64..5.5] [8.2..14.5] [15.67..56.33]
Performance_estimation	[73..87] [90..95] [100..100]	Example_rule_ratio	[4.69..16.29] [18.45..56.3]
Last_simplicity	[3..15] [19..46] [48..155]	Training_examples_ percent_of_total	[0] [5.5..39]

DCI-Gen if:

[Relative_rule_weight1 = 0.62..0.99] & [Training_examples_percent_of_total = 5.5..39] OR

[Performance_estimation = 73..87] & [Example_rule_ratio = 4.69..16.29]

Interpretation:

Perform DCI Generation of attributes if the second disjunct in every rule covers almost as many examples as the heaviest rule, and the training set represents a high percentage of the total possible space; or if the predictive accuracy of the last run was relatively low and, individually, the rules cover a relativly small number of examples.

These rules are stating that DCI-Gen is useful if the concepts are blurred across attributes. This occurs when the coverage of learned rules is low across most rules.

DCI-Select if:

[Last_simplicity = 48..155] & [Avg. #_of_uniquely_covered_examples = 8.2..14.5] & [Example_rule_ratio = 4.69..16.29]

Interpretation:

Perform DCI Selection of attributes if the last set of rules learned had a high number of selectors, and the learned rules had a moderate degree of overlap, and and, individually, the rules cover a relativly small number of examples.

This rule is saying that DCI-Select is useful when the simplicity of the learned rules is high and there are no strong heavy rules learned.

AQ if:

[Last_simplicity = 3..15]

Interpretation:

Perform no change to the representation space if the learned rules are simple.

DCI-Quant if:

[Last_simplicity = 19..46] & [Training_examples_percent_of_total = 0]

Interpretation:

Perform DCI-Quant abstraction of the space if the last learned rules were moderately complex and the training examples provided is only a small percentage of the total possible space. This latter condition is clearly common when the size of the attribute domains are large.

HCI-Gen if:

[Number_of_attributes = 5..28] & [Selector_rule_ratio = 5..9.64]

Interpretation:

Perform HCI generation of attributes if the number of attributes is relatively low and the rules contain a large number of conditions.

The predictive accuracy of the learned rules for the 6 classes was 7/14 or 50%. This is significantly better than random guessing (1/6 or 16%). When we evaluate what types of mistakes were made, it can be seen that the rules were working even better. A detailed description of the seven mistakes are given below. This shows that although the RSM operator selected was not the best possible, an RSM operator which could improve the RS was selected 10 of the 14 (71%) of the time.

In the second experiment the available RSM operators have been placed into only three classes for meta-rule purposes:

1. No modification (AQ only)
2. Representation Space Expansion (DCI-, HCI-GENERATE))
3. Representation Space Contraction (DCI, HCI-SELECT, and DCI-QUANT)

With learning from these fewer, hierarchically ordered classes, the learned meta-rules predictied the best RSM operator for 12 of the 14 examples (86%). Mistakes were made on the Monk #1 problem when RS contraction was selected instead of expansion and for T1.10 when contraction was suggested instead of the correct choice of no change (Figure 5.15). Interestingly, the T1.10 problem clearly needs some kind of attribute selection because of its many low quality attributes, but because there are so few training examples, both DCI and HCI with their default settings result in learned rules which perform worse than the rules learned in the original space.

Table 5.15 - Descriptions of RSM operator selection errors

Problem	Best RSM	Best Accuracy	Meta-rule selected RSM	Accuracy	Original Accuracy
Security	DCI-SCALE	96%	DCI-ADD	87%	89%
T1.40	HCI-SEL	93%	DCI-SELECT	90..95%	91%
Scaled_hepatitis	HCI-GENERATE	97%	DCI-GENERATE	95%	95%
Hepatitis	DCI-GENERATE	97%	HCI-GENERATE	86%	73%
Monk4	DCI-SELECT	93%	DCI-GENERATE	91%	93%
T1.20	DCI-SELECT	89%	DCI-GENERATE	88%	90%
T1.60	DCI-SELECT	97%	HCI-SELECT	92%	92%

In summary, learned meta-rules can select an RSM operator that improves the representation space with a high degree of accuracy.

5.5.3 Meta-Rule Evaluation: Incremental

This section describes an experiment designed to evaluate how the incremental addition of new examples affected the performance of learned meta-rules. More specifically the goal of this experiment is to determine how many times the meta-rule controller must quess before making the correct choice. The dataset used for this experiment was the same as in Section 5.5.2. The first step was to randomly assign an example for every meta-class: a rule cannot be learned, and a class cannot be predicted if at least one example is not given. With these six examples removed for training, the testing began. Each of the remaining eight examples were tested against the current rules, giving a degree of match between that example and all six classes. If the correct class matched the training example then the assigned rank was 0. If the correct class was the second strongest matching class, then the assigned rank was 1. The possible ranks ranged from 0 to 5. Perfect meta-rules always predict the correct class right away and would average a rank of 0 for every class. An exhaustive approach would average a rank of 5 because every operator (represented by the six classes) would be tried. If learned meta-rules are indeed improving over time, then the rank for a class will decrease as new examples are made available for training. Examples are indexed from 1 to 14.

In the first experiment examples 10 (Hepatitis), 6 (t1.60), 9 (Security), 14 (Scale-Hepatitis), 11 (t1.40) and 13 (M2DCI) were assigned to the classes DCI-Gen, DCI-Scale, DCI-Quant, HCI-Gen, HCI-Sel and AQ respectively. Characteristic meta-rules were learned from these examples. The remaining eight examples were then tested against the learned rules, assigned a rank and then added to the appropriate class. The results of these tests are shown in Table 5.16. Class rank is shown by "/". Those classes with the same degree of match have equal rank.

Table 5.16 shows that even with very few examples the meta-rules perform quite well. The best class was selected first (rank=0) for 3 of the last five test cases. With so little data it is hard to draw strong conclusions, but these results are promising. It is also interesting to see the change broken down by class. DCI-Quant had rank 2 after 1 meta-example, but predicted the class of t3.60 correctly right away when it had two meta-examples. Similarly AQ went from rank 4 to rank 0, and DCI-Sel went from rank 4 to rank 0. The ranks of DCI-Gen and HCI-Gen did not improve probably because the initial example was so different than the second example (Hepatitis to Monk1) and (Scale-Hepatitis to Monk2).

Table 5.16 - Predicted Ranks for learned meta-rules: experiment 1

Index	Name	Correct Class	Rank	Class order by degree of match
8	t340	DCI-Quant	3	DCI-Sel/HCI-SEl/DCI-Quant
1	Monk1	DCI-Gen	5	AQ/DCI-Quant/DCI-Sel/HCI-Sel/HCI-Gen/DCI-Gen
4	t1.10	AQ	4	HCI-Sel/DCI-Sel/DCI-Quant/DCI-Gen/AQ
12	M1-DCI	AQ	0	AQ/DCI-Gen/DCI-Quant/DCI-Sel-HCI-Sel/HCI-Gen
7	t3.60	DCI-Quant	0	DCI-Quant/DCI-Sel-AQ-HCI-Sel/DCI-Gen/HCI-Gen
5	t1.20	DCI-Select	4	AQ-HCI-Sel/DCI-Gen-DCI-Quant/DCI-Sel/HCI-Gen
2	Monk2	HCI-Gen	5	DCI-Sel/DCI-Gen/AQ/DCI-Quant/HCI-Sel/HCI-Gen
3	Monk4	DCI-Sel	0	DCI-Sel/DCI-Quant/AQ-HCI-Sel/DCI-Gen-HCI-Gen

In the second experiment examples 1 (Monk1), 5 (t1.20), 9 (Security), 2 (Monk2), 11 (t1.40) and 4 (t1.10) were assigned to the classes DCI-Gen, DCI-Scale, DCI-Quant, HCI-Gen, HCI-Sel and AQ respectively. Characteristic meta-rules were learned from these examples. The remaining eight examples were then tested against the learned rules, assigned a rank and then added to the appropriate class. The results of these tests are shown in Table 5.17.

Although the average for this experiment is slightly worse than for the previous experiment, some individual classes still show improvement. DCI-Sel improved from rank 1 to rank 0, and even more dramatically AQ improved from rank 5 to rank 0. DCI-Quant increased in rank from 0 (for t3.60), to 3 for test problem t3.40. The smaller amount of training data in the second example apparently mislead it the meta-rules to select DCI-Sel and HCI-Sel before DCI-Quant. HCI-Gen and DCI-Gen had the same problems as in the previous test case- the examples used in the previous iteration were very different than those used in the next iteration. This time the training cases were Monk1 and Monk2 and the test cases were Hepatitis and Scale-Hepatitis respectively.

Table 5.17 - Predicted Ranks for learned meta-rules: experiment 2

Index	Name	Correct Class	Rank	Class order by degree of match
7	t3.60	DCI-Quant	0	DCI-Quant/AQ-DCI-Sel-HCI-Sel/DCI-Gen-HCI-Gen
6	t1.60	DCI-Sel	1	HCI-Sel/DCI-Sel-AQ/DCI-Quant/DCI-Gen-HCI-Gen
14	Scale-Hep	HCI-Gen	5	DCI-Sel/AQ-HCI-Sel/DCI-Quant/DCI-Gen-HCI-Gen
3	Monk4	DCI-Sel	0	DCI-Sel/DCI-Quant/AQ-HCI-Sel/HCI-Gen/DCI-Gen
8	t3.40	DCI-Quant	3	DCI-Sel/AQ-HCI-Sel/DCI-Quant/HCI-Gen/DCI-Gen
10	Hepatitis	DCI-Gen	5	DCI-Sel/HCI-Gen/HCI-Sel/AQ/DCI-Quant/DCI-Gen
12	M1DCI	AQ	5	DCI-Sel/DCI-Gen/DCI-Quant-HCI-Gen/HCI-Sel-AQ
13	M2DCI	AQ	0	AQ/DCI-Sel/DCI-Gen/DCI-Quant-HCI-Add/HCI-Sel

The section described four different experiments designed to evaluate the effectiveness of learning meta-rules for selecting the best representation space modification operator. The first two experiments used a traditional leave-one-out methodology and showed with very few meta-examples the best operator was selected 50% of the time and an operator that improved the space was selected 71% of the time. Furthermore, when the RSM operator classes are organized hierarchically the best class was selected 86% of the time. The second set of experiments showed that the ability to predict the best class does improve as more examples are provided and that this improvement can be seen after only two examples of a class have been seen.

CHAPTER 6 CONCLUSIONS

6.1 Summary

This thesis presents an approach to multistrategy constructive induction (MCI) in which multiple representation space modification operators are used to transform difficult problems into easy problems. This work showed that MCI can significantly improve the quality of learned hypotheses for a wide variety of problems, MCI can be composed of RSM operators using different computational strategies (statistics, evolutionary compuation, heuristics), and that the control rules (meta-rules) which select RSM operators can be learned from past experience with a high degree of accuracy.

Additionally[viii], this dissertation introduces a novel methodology for multistrategy constructive induction. The approach 1) incorporates multiple computational methods for constructive induction including representation space expansion and contraction, 2) incorporates multiple inferential techniques in its use of deduction to arrive at a meta-decision concerning which representation space modifier to select, and induction when inducing a new (or modifying a previous) meta-rule from a set of meta-examples, and 3) it is a learning system capable of improving its own performance over time through meta-learning. The proposed method is built on established individual empirical induction and constructive induction techniques and is capable of incorporating knowledge from many sources including a) directly from the user, b)

[viii]Some of this section is a reiteration of section 1.6.

from analysis of the data, and c) from analysis of learned hypotheses.

The proposed multistrategy approach helps to overcome the brittleness of current learning methods by automating the search for representation spaces which are better suited to learning predictively accurate rules. This approach helps overcome the problems such methods have with complex real-world data. Continuous-valued data problems are overcome by including a method for attribute-value discretization. Misclassification noise can be corrected by attribute selection. Poorly represented attributes can be improved with attribute generation with DCI-Gen or HCI-Gen. The meta-learning capabilities eliminate the need for human expertise to guide the selection of these tools. The relationship between characteristics of a dataset and appropriate representation space transformations are not generally known. A learning approach to this meta-learning task eliminates the need to explicitly determine this relationship before using the available tools. Such meta-rules also are helping to discover general principles of inductive learning currently unknown or poorly understood. However, as each learning example is a dataset, this learning requires data which is not available in great quantities. Additional experiments with significantly more data are required to find meta-learning trends for which there is high confidence.

6.2 Future Work

Although quite powerful already, there are a number of additions and improvements that can be made to AQ17-MCI. These additions focus on the control strategy and the available representation space modification operators.

The control strategy currently performs a 1-fold knowledge-driven greedy search. If a modification results in an improvement the space is changed and search continues. However, it may occur that the best set of transformations necessary to correct a problem do not result in improvements at every step. A greedy approach with no look-ahead will not find combinations of operators that do not improve the space at every step. One way to overcome this limitation is to see which combinations of operators often occur together and bundle them as one operator. For example abstraction followed by attribute-generation can result in a large gain in performance. If performed in one step, the greedy control method cannot separate the two steps even if one alone results in a decrease in performance. The control strategy is also restricted to controlling only the data-driven and hypothesis-driven operatrors. Those transformations suggested by the user are automatically made before any other modification. In order to encourage learning across problems, the knowledge acquired from past problems should be stored and used for future problems. AQ17-MCI may make use of a goal-dependency network (GDN) (Michalski and Stepp, 1986) to record knowledge about domains and past successes in order make better use of user-given knowledge.

The available operators in the toolbox of AQ17-MCI can also be expanded and improved. As described in chapter 3, there are a number of other attribute construction methods, that make use of other inferential transmutations, which could be included: a) attribute construction using a genetic algorithm, or b) analogy based meta-learning for acquiring better meta-rules. The current operators can also be improved. The current abstraction operator uses chi-merge. This method does not produce hierarchies of ordered attributes, nor does it take into account the abstraction of other attributes. This is a problem for any algorithm that only looks at the data one attribute at a time. Linear hierarchies could result in even better descriptions of the available data because it would give to AQ the detail (precision) of the lowest level when it needed it, while also providing the abstractions of large intervals to use when describing areas within a class.

REFERENCES

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

Baim, P.W., "The PROMISE Method for Selecting Most Relevant Attributes for Inductive Learning Systems", Reports of the Intelligent Systems Group, Department of Computer Science, University of Illinois at Urbana-Champaign, ISG 82-5, 1982.

Bala, J., "Learning to Recognize Visual Concepts: Development and Imnplementation of a Method for Texture Concept Acquisition Through Inductive Learning", *Ph.D. dissertation*, School of Information Technology and Engineering, *Reports of the Machine Learning and Inference Laboratory*, MLI93-3, Center for Artificial Intelligence, George Mason University, March 1993.

Banerji, R.B., "Learning in the Limit in a Growing Language," *Proceedings of IJCAI-87*, pp. 280-282, Milan, Italy, 1987.

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

Belyaev, L., Falcone, L.P., "Noise-Resistant Classification," *Proceedings of the Eighth International Workshop on Machine Learning*, pp. 581-585, Evanston, Ill., 1991.

Bloedorn, E., and Kaufman, K., "Data-Driven Constructive Induction in INLEN," Reports of the Machine Learning and Inference Laboratory, George Mason University, 1996.

Bloedorn, E., Michalski, R.S. and Wnek, J., "Matching Methods to Problems," George Mason University, Machine Learning and Inference Laboratory, 1994.

Bloedorn, E., and Michalski, R.S., "Constructive Induction from Data in AQ17-DCI: Further Experiments," Center for Artificial Intelligence, George Mason University, MLI 91-12, 1991.

Bloedorn, E., "Constructing Problem Relevant Attributes for Classification Using a Genetic Algorithm: CI-GA," *Report for INFT 910: Advanced Topics in AI: Evolutionary Computation*, George Mason University, 1993.

Bloedorn, E., "Goal-Driven Property Transfer: An Implementation and Application," *Class-Report*, George Mason University, Report for CS-785 Knowledge Acquisition, 1993.

Bloedorn, E., and Michalski, R.S., "The AQ17-DCI System and it Application to World Economics," *Proceedings of the Ninth International Symposium on Methodologies for Intelligent Systems*, pp. Zakopane, Poland, 1996.

Brazdil, P., Gama, J. and Henery, R., "Characterizing the Applicability of Classification Algorithms using Meta Leval Learning", in *Machine Learning-ECML-94,* F., Bergadano, and L. de Raedt (Eds.), Springer Verlag. 1994.

Brodley, C., "Addressing the Selective Superiority Problem: Automatic Algorithm/Model Class Selection," *Proceedings of the Tenth International Conference on Machine Learning*, pp. 17-24, 1993.

Callan, J.P., and Utgoff, P.E., "Constructive Induction on Domain Information," *Proceedings of AAAI-91*, pp. 614-619, 1991.

Callan, J.P., Utgoff, P.E., "A Transformational Approach to Constructive Induction," *Proceedings of the Eight International Workshop on Machine Learning*, pp. 122-126, Evanston, Ill., 1991.

Carbonell, J.G., Michalski, R.S. and Mitchell, T.M., "An Overview of Machine Learning," *Machine Learning: An Artificial Intelligence Approach*, Ryszard S. Michalski, Jaime G. Carbonell and Tom M. Mitchell (Eds.), Morgan Kaufmann, 1983.

Caruana, R., "Multi-task Learning: A Knowledge-Based Source of Inductive Bias", *Proceedings of the Tenth International Conference on Machine Learning*, p. 41-49, Amherst, MA, 1993.

Catlett, J., "On Changing Continuous Attributes Into Ordered Discrtete Attributes", in Y. Kodratoff (Ed.) *Proceedings of the European Working Session on Learning*, Berlin, Germany, Springer-Verlag, pp. 164-178, 1991.

Clarke, P., and Reichgelt, H., "Explorations in Representation Change," *Proceedings of IJCAI-91 Workshop on Evaluating and Changing Representation in Machine Learning*, pp. 90-95, Sydney, Australia, 1991.

Cohen, W.W., "An Analysis of Representation Shift in Concept Learning," *Proceedings of the 7th International Conference on Machine Learning*, pp. 104-112, Austin, TX, 1990.

Dabija, V., Tsujino, K., and Nishida, S., "Learning to Learn Decision Trees", *Proceedings of AAAI-94*, p. 88-95, 1994.

Donoho, S., and Rendell, L., "Representing and Restructuring Domain Theories: A Constructive Induction Approach", *Journal of Artificial Intelligence Research*, Vol. 2., pp. 411-446.

Donoho, S. and Rendell, L., "Constructive Induction Using Fragmentary Knowledge", *Proceedings of the Thirteenth International Conference on Machine Learning*, Bari, Italy, pp. 113-121, 1996.

Dougherty, J., Kohavi, R., and Sahami, M., "Supervised and Unsupervised Discretization of Continuous Features", *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 194-202. San Francisco, 1995.

Drastal, G., Czako, G., and Raatz, S., "Induction in an Abstraction Space: A Form of Constructive Induction," *Proceedings of IJCAI-89.* pp. 708-712, 1989.

Elio, R., and Watanabe, L, "An Incremental Deductive Strategy for Controlling Constructive Induction in Learning from Examples", *Machine Learning*, Vol. 7, p. 7-44. Kluwer Academic, 1991.

Fayyad, U., Irani, K., "Multi-internval discretization of continuous-valued attributes for classification learning", *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, pp. 1022-1027, 1993.

Fulton, T., Kasif, S., and Salzberg, S., "Efficient Algorithms for Finding Multi-way Splits for Decision Trees", *Proceedings of the Twelfth International Conference on Machine Learning (ML95)*, Armand Preditis andStuart Russel (Eds.), Morgan Kaufmann, p. 244-251, Tahoe City, CA, July 1995.

Genest, J., Matwin, S. and Plante, B., "Explanation-Based Learning with Incomplete Theories: A Three-step Approach," *Proceedings of the 7th International Conference on Machine Learning*, pp. 286-294, Austin, TX, 1990.

Gordon, D., *Active Bias Adjustment for Incremental Supervised Concept Learning*, Department of Computer Science, University of Maryland, Ph.D, 1990.

Gordon, D.F., and Desjardins, M., "Evaluation and Selection of Biases in Machine Learning," *ML*, 20, Vol. pp. 1995.

Graner, N., Sharma, S., Sleeman, D. et al., "The Machine Learning Toolbox," Department of Computer Science, University of Aberdeen, AUCS/TR9207, 1992.

Greene, G., "Quantitative Discovery: Using Dependencies to Discover Non-Linear Terms", *Masters Thesis*, University of Illinois, Urbana-Champaign, 1988.

Gregg, M., Gunsch, H., Rendell, L.A., "Opportunistic Constructive Induction," *Proceedings of the Eight International Workshop on Machine Learning*, p. 147-152, Evanston, Ill., 1991.

Hong, J., Mozetic, I., and Michalski, R. S., "AQ15: Incremental Learning of Attribute-Based Descriptions from Examples, the Method and User's Guide," *Report ISG 86-5, UIUCDCS-F-86-949*, p. Computer Science Dept., University of Illinois at Urbana-Champaign, 1986.

Hu, Y., and Kibler, D., "Generation of Attributes for Learning Algorithms", *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI96),* p. 806-811, Portland, OR, 1996.

Imam, I., Michalski, R.S. and Kerschberg, L., "Integrating Machine Learning and Statistical Techniques for Mining in Databases," *submitted to the First International Workshop on Knowledge Discovery in Databases*, pp. Washington, D.C., 1993.

Imam, I., "Determing Attribute Relevancy", *Proceedings of the Ninth International Symposium on Methodologies for Intelligent Systems*, pp. Zakopane, Poland, 1996.

John, G., Kohavi, R., and Pfleger, K., "Irrelevant Features and the Subset Selection Problem", *Proceedings of the Eleventh International Conference on Machine Learning (ML94),* Morgan Kaufmann, pp. 121-129.

Kaufman, K., "Comparing International Development Patterns Using Multi-Operator Learning and Discovery Tools," *Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases*, pp. 431-440, Seattle, WA, 1994.

Kelly, G.A., *The Psychology of Personal Constructs*, Norton, New York, 1955.

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

Kira, K., and Rendell, L., "The Feature Selection Problem: Tradtional Methods and a New Algorithm", In Proceedings of AAAI-92, pp. 129-134, MIT Press, 1992.

Kokar, M., "Discovering Functional Formulas through Changing Representation Base", Proceedings of AAAI-86, p. 455-459, Philadelphia, PA, 1986.

Koller, D., and Sahami, M., "Toward Optimal Feature Selection", in Proceedings of the Thirteenth International Conference on Machine Learning (ML96), p. 284-292, 1996.

Kramer, Stefan, "CN2-MCI: A Two-Step Method for Constructive Induction", Proceedings of the Eleventh International Conference on Machine Learning (ML94), p. 33-39. 1994.

Kubat, M., "Second Tier for Decision Trees", *Proceedings of the Thirteenth International Conference on Machine Learning,* p. 293-301, Bari, Italy, 1996.

Langley, P., Bradshaw, G., and Simon, H., "Rediscovering Chemistry with the BACON System", In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann, Los Altos, CA, 1983.

Langley, P., Simon, H., Bradshaw, G., and Zytkow, J., "Scientific Discovery: Computational Explorations of the Creative Processes", MIT Press, Cambridge, MA, 1987.

Ling, X., and Ungar, L., "Inventing Theoretical Terms in Inductive Learning of Functions - Search and Constructive Methods," *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*, pp. 332-341, 1989.

Liu, H., and Setiono, R., "A Probabilistic Approach to Feature Selection - A Filter Solution", Proceedings of the Thirteenth International Conference on Machine Learning (ML96), p. 319-327, Bari, Italy, 1996.

Matheus, C., *Feature Construction: An Analytic Framework and an Application to Decision Trees*, University of Illinois at Urbana-Champaign, Ph.D, 1990.

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

Matheus, C.J., "Adding Domain Knowledge to SBL through Feature Construction," *Proceedings of AAAI-90*, pp. 803-808, Boston, MA, 1990.

Matheus, C.J., "A Constructive Induction Framework," *Proceedings of the 6th International Workshop on Machine Learning*, pp. 474-475, Ithaca, NY, 1989.

Matwin, S., and Plante, B., "A Deductive-Inductive Method for Theory Revision," *Machine Learning: A Multistrategy Approach,* R.S. Michalski and G. Tecuci (Eds.), San Mateo, CA: Morgan Kaufmann Publishers, 1992.

Mehra, P., "Constructing Representations Using Inverted Spaces," *Proceedings of the 6th International Workshop on Machine Learning*, pp. 472-473, Ithaca, NY, 1989.

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

Michalski, R.S., "Pattern Recognition as Knowledge-Guided Computer Induction," (Technical Report No. 927). Department of Computer Science, University of Illinois, Urbana-Champaign, IL, 1978.

Michalski, R.S., "A Theory and Methodology of Inductive Learning," *Artificial Intelligence*, Vol. pp. 1983.

Michalski, R.S., Mozetic, I., Hong, J., Lavrac, N., "The multi-purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains", Proceedings of AAAI-86, p. 1041-1045, Philadelphia, PA, 1986.

Michalski, R.S., "Two-Tiered Concept Mearning, Inferential Matching, and Conceptual Cohesiveness", In Similarity and Analogical Reasoning, S. Vosniadou, and A. Ortony (Eds.), p. 122-145, Cambridge University Press, 1989.

Michalski, R.S., and Watanabe, L.M., "Constructive Closed-Loop Learning: Fundamental Ideas and Examples," George Mason University, Reports of Machine Learning and Inference Laboratory MLI-88-1, 1988.

Michalski, R.S., "Inferential Learning Theory as a Basis for Multistrategy Task-Adaptive Learning," *Proceedings of the First International Workshop on Multistrategy Learning*, pp. 3-18, Harpers Ferry, WV, 1991.

Michalski, R.S., "Toward a Unified Theory of Learning: An Outline of Basic Ideas," *Invited presentation at the First World Conference on the Fundamentals of Artificial Intelligence*, pp. Paris, France, 1991.

Michalski, R.S., "Inferential Theory of Learning: Developing Foundations for Multistrategy Learning," GMU Center for AI, MLI 92-03, 1992.

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

Michalski, R.S., and Ram, A., "Learning as Goal-Driven Inference", *Reports of the Machine Learning and Inference Laboratory*, MLI94-26, George Mason University, Fairfax, VA, July 1994.

Michie, D., Spiegelhalter, D.J. and Taylor, C., *Machine Learning, Neural and Statistical Classification*, Ellis Horwood, 1994.

Mitchell, T., "The Need for Biases in Machine Learning," Rutgers University, CBM-TR-117, 1980.

Mohan, S., and Tong, C., "Automatic Construction of a Hierarchical Generate-and-Test Algorithm," *Proceedings of the 6th International Workshop on Machine Learning*, pp. 483-484, Ithaca, NY, 1989.

Pagallo, G., and Haussler, D., "Boolean Feature Discovery in Emprical Learning", Machine Learning Vol. 5, p. 71-99, Kluwer Academic, 1990.

Pagallo, G., "Adaptive Decision Tree Algorithms for Learning from Examples", PhD Thesis, University of California at Santa Cruz, June, 1990.

Perez, E., and Rendell, L., "Using Multidimensional Projection to Find Relations", *Proceedings of the Twelfth International Conference on Machine Learning*, p. 447-455, Morgan Kaufmann, 1995.

Pfahringer, B., "Compression-Based Discretization of Continuous Attributes", *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 456-463. San Francisco, 1995.

Provost, F., J., *Policies for the Selection of Bias in Inductive Machine Learning*, University of Pittsburgh, Ph.D, 1992.

Quinlan, J.R., *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann, 1993.

Quinlan, J.R., "Improved Use of Continuous Attributes in C4.5", *Journal of Artificial Intelligence Research*, Vol. 4., pp. 77-90, 1996.

Ragavan, H., and Rendell, L., "Lookahead Feature Construction for Learning Hard Concepts", Proceedings of the Tenth International Conference on Machine Learning (ML93), 1993.

Rao, R., Gordon, D., and Spears, W., "For Every Generalization Action is there Really an Equal and Opposite Reaction? Analysis of the Conservation Law for Generalization Performance", *Proceedings of the Twelfth International Conference on Machine Learning*, p. 471-479, Morgan Kaufmann, 1995.

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

Rendell, L., Seshu, R. and Tcheng, D., "More Robust Concept Learning Using Dynamically-Variable Bias" *Proceedings of the Fourth International Machine Learning Workshop*, pp. 66-78, Irvine, CA, 1987.

Rummelhart, D.E., and McClelland, J.L. (Eds.), *Parallel Distributed Processing: Psychological and Biological Models*, MIT Press, 1986.

Schaffer, C., "A Conservation Law for Generalization Performance", *Proceedings of the Eleventh International Conference on Machine Learning*, p. 259-265, Morgan Kaufmann, 1994.

Schlimmer, J.C., *Concept Acquisition through Representational Adjustment*, University of California, Irvine, 1987.

Sutton, and Matheus, C., "Learning Polynomial Functions by Feature Construction", Proceedings of the Eighth International Workshop on Machine Leanring (ML91), p. 208-212, 1991.

Tecuci, G., and Kodratoff, Y., *Apprenticeship Learning in Imperfect Domain Theories*, in Machine Learning: An Artificial Intelligence Approach, Vol III, Y. Kodratoff and R.S. Michalski (Eds.), Morgan Kaufmann, Los Altos, CA, 1990.

Thrun, S., and O'Sullivan, J., "Discovering Structure in Multiple Learning Tasks: The TC Algorithm", *Proceedings of the Thirteenth International Conference on Machine Learning (ML96)*, p. 489-597, Morgan Kaufmann, Bari, Italy, 1996.

Utgoff, P., "Shift of Bias for Inductive Concept Learning," *Machine Learning: An Artificical Intelligence Approach*, R.S. Michalski, J. Carbonnel and T. Mitchell (Eds.), San Mateo, CA: Morgan Kaufman, 1986.

Vafaie, H., and De Jong, K., Improving the Performance of a Rule Induction System Using Genetic Algorithms", in Machine Learning: A Multistrategy Approach", Vol. IV, R.S. Michalski, and G. Tecuci (Eds.), Morgan Kaufmann, San Mateo, CA, 1994.

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

Weiss, S.M., and Kulikowski, C.A., *Computer Systems that Learn*, San Mateo, CA: Morgan Kaufmann, 1991.

Wnek, J., and Michalski, R.S., "Hypothesis-Driven Constructive Induction in AQ17 - A Method and Experiments," *Proceedings of IJCAI-91 Workshop on Evaluating and Changing Representation in Machine Learning*, pp. 13-22, Sydney, Australia, 1991.

Wnek, J., "Hypothesis-driven Constructive Induction", *Ph.D. dissertation*, School of Information Technology and Engineering, *Reports of the Machine Learning and Inference Laboratory*, MLI93-8, Center for Artificial Intelligence, George Mason University, March 1993.

Wnek, J., "User Manual: Specification and Guide through the Diagrammatic Visualization System," George Mason University, Machine Learning and Inference Laboratory, MLI95-5, 1995.

Wnek, J., and Michalski, R.S., "Hypothesis-driven Constructive Induction in AQ17-HCI: A Method and Experiments," *Machine Learning*, 14, pp. 139-168, Vol. p. 1994.

Wnek, J., Kaufman, K., Bloedorn, E., and Michalski, R.S., "Selective Inductive Learning Method AQ15c: The Method and User's Guide", *Reports of the Machine Learning and Inference Laboratory*, MLI95-4, George Mason Univerity, Fairfax, VA, 1995.

Wrobel, S., "Demand-Driven Concept Formation," *Knowledge Representation and Organization in Machine Learning*, K. Morik (Eds.), Berlin Heidelberg: Springer Verlag, 1989.

Yang, Der-Shung, Rendell, L., and Blix, G., "A Scheme for Feature Construction and a Comparison of Empirical Methods", Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, Sydney, Australia, 1991.

Yip, S., and Webb, G., "Incorporating Canonical Discriminant Attributes in Classification Learning", *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, p. 63-70, Morgan Kaufmann, 1994.

Zembowicz, R., and Zytkow, J., "Automated Discovery of Empirical Equations from Data", *Proceedings of ISMIS91*, Z. Ras (Ed.), Springer-Verlag, 1991.