MATCHING METHODS WITH PROBLEMS: A COMPARATIVE ANALYSIS OF CONSTRUCTIVE INDUCTION APPROACHES

by

E. Bloedorn
R. S. Michalski
J. Wnek

Reports of the Machine Learning and Inference Laboratory, MLI 94-2, School of Information Technology and Engineering, George Mason University, Fairfax, VA, May 1994.

MATCHING METHODS WITH PROBLEMS:
A Comparative Analysis of
Constructive Induction Approaches

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

MLI 94-2

May 1994

Matching Methods with Problems: A Comparative Analysis of Constructive Induction Approaches

Abstract

This paper provides a taxonomy of constructive induction problems and reports on an empirical comparison of several constructive induction methods. In this paper a representation space is said to be poorly suited for learning because of three types of problems: 1) inappropriate attributes or attribute values, 2) incomplete attribute values, attribute sets or examples and 3) incorrect attributes or examples. Most current constructive induction methods are designed to correct one of these types (or sub-types) of problems which limits the types of problems for which this method is effective. In order to build a more general multistrategy method of constructive induction an understanding of when some methods for constructive induction are useful and when they fail is important. Five methods of constructive induction are evaluated: DCI attribute construction (AQ-DCI), HCI attribute construction (AQ-HCI(ADD)), HCI attribute removal (AQ-HCI(REMOVE)), HCI construction and removal (AQ-HCI), and attribute-value removal (AQ-SCALE). The results point to the need for a multistrategy constructive induction approach for solving a wide variety of induction problems.

Key words: concept learning, constructive induction, multistrategy learning,

Acknowledgements

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

1. Introduction

The underlying idea of constructive induction (CI) is to view learning as a double search process. This contrasts with standard machine learning which performs only a single search. CI methods perform a search both for an 'adequate' representation space, and for a hypothesis within that space.

By representation space is meant a space in which facts, hypotheses and background knowledge are represented. The representation space is spanned over descriptors that are elementary concepts used to characterize examples from some viewpoint. Usually examples are given as vectors of single argument descriptors (attributes). In this paper the discussion will be limited to an attributional representation. 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 VL1), hyperplanes or higher degree surfaces (neural nets), and compositions of elementary structures (grammars).

Both the search for an adequate representation space and the search for a hypothesis within that space are performed through the repeated application of available search operators. The search for a hypothesis applies operators provided by the given inductive learning method. For example, the AQ17-MCI method (Bloedorn, Michalski and Wnek, 1993) uses operators employed in the AQ-type learning systems, such as "dropping conditions," "extension against," "adding an alternative," "closing interval," and "climbing a generalization tree." The representation space operators modify the representation space. These can generally be classified into "expanders," that expand the space by adding new dimensions (attributes), and "contractors" that contract the space by removing less relevant attributes and/or abstracting values of some attributes.

Since the inception of this view of CI as a double search, several methods for CI have been proposed (Wnek and Michalski, 1994, Wnek and Michalski, 1994). However, a comparison of the relative strengths of various methods has not been performed. This paper performs a comparison for five different CI methods: AQ-HCI(ADD), AQ-HCI(REMOVE), AQ-HCI (ADD and REMOVE combined), AQ-DCI and AQ-SCALE. This analysis shows a strong need for multistrategy CI in order to extend the range of problems performed by empirical induction.

2. Related Work

The goal of constructive induction is to find simple, accurate descriptions of the concepts to be learned. One approach to finding such descriptions is to search for a induction method that was well suited to the distribution of examples. This approach is taken by Brodley (Brodley, 1993) in which a number of different methods, each with different type of constructs for building descriptions, are combined. The selection of an appropriate induction method is based on a number of heuristic rules.

Work on the automated selection an appropriate bias is also closely related to this research. Bias-adjustment programs include STABB (Utgoff, 1986) and VBMS (Rendell, Seshu and Tcheng, 1993). STABB searches for a bias by searching for a better description language. STABB can both weaken the bias (expand the representation space) and strengthen the bias (contract the representation space). STABB does not, however, learn when certain bias modifications are useful. The VBMS system does associate biases (different induction algorithms) with problem characteristics using a similarity-based clusterer PLS1. VBMS did not, however, have any mechanism for modifying the representation space by adding or removing terms.

Current methods for CI are limited because only one method for modifying the representation space has been used. Real problems often have multiple difficulties associated with them such as misclassified examples *and* a number of irrelevant attributes. A system that combines multiple

methods for performing constructive induction has the potential for greatly extending the range of problems susceptible to empirical induction. Introducing multiple tools, however, introduces a number of questions of its own: Which methods should be used together? When should each available method be used? Are there results complementary or destructive?

3. A Taxonomy of Representation Space Modifiers

The operators for searching for a hypothesis in a given a representation space are well understood (Michalski, 1983), but the search operators for finding a good representation are less well understood. In order to develop a taxonomy of constructive induction methods it is useful to examine the assumptions made by selective induction methods because it is these assumptions that are violated by real-world problems.

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. 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. For example, a system will not be able to generate accurate useful rules for medical diagnosis if it has no information concerning the physical condition of the patient. 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 noise, but it can also be due to error. Selective induction methods assume that the given data are in an appropriate form so that examples which are close to each other in the representation space are also close to, or identical in class membership as well (Rendell and Seshu, 1990). When any of these assumptions are violated the representation space is inadequate for selective induction and poor descriptors (low predictive accuracy and high complexity) result.

In this paper CI is viewed as a double search process. Some of the inadequacies raised here can be corrected by changing the constructs in the hypothesis language *or* by changing the representation space itself. In this research the hypothesis language is assumed to be fixed. Therefore, any modifications to make the learning problem easier must come about by changing the representation language. Thus, in this discussion, the emphasis is on finding an appropriate representation space given a single hypothesis language. The VL_1 hypothesis language was selected because it is comprehensible and powerful.

A representation space is inadequate when it is either incomplete, incorrect or inappropriate (Fig. 1). The role of the representation space modifiers in constructive induction is to correct the inadequacies of the representation space so that good hypotheses can be found. To achieve a learning method which can overcome any of the deficiencies of the representation space methods for correcting each of these inadequacies must be developed.

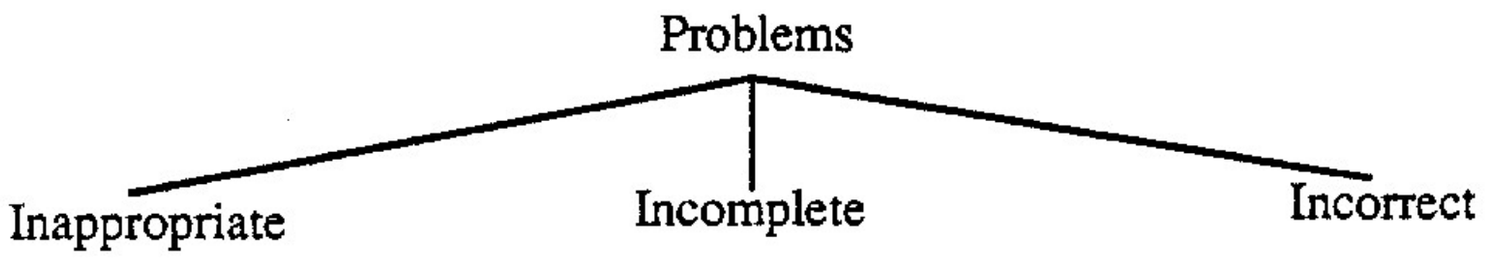

Figure 1. A taxonomy of representation space inadequacies

3.1 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.

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. A value set that contains overly precise values, however, 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 2.1). 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. Various methods for automatic discretization of attribute data have been proposed. Some of these methods are quite simple such as equal-width intervals, and equal-frequency intervals. Others such as C4.5 (Quinlan, 1993), and SCALE which implements the Chi-merge algorithm (Kerber, 1992) are more complex.

Inappropriate attributes are those attributes which are relevant to the problem at hand, but which pose the problem in such a way that the descriptive constructs of the language are inadequate. For example, the parity problem when stated in terms of the presence of 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). Some representative of each of these types are: AQ17-DCI (Bloedorn and Michalski, 1991), BLIP (Wrobel, 1989), CITRE (Matheus and Rendell, 1989), Pagallo and Haussler's FRINGE, GREEDY3 and GROVE (Pagallo and Haussler, 1990), MIRO (Drastal, Czako and Raatz, 1989) and STABB (Utgoff, 1986).

3.2 Incompleteness

An induction problem is incomplete if attribute values, attributes (concepts), 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, 1990). In DISCIPLE machine learning methods are used to guide the questioning of the expert to most efficiently fill gaps in the knowledge base.

A problem may also be incomplete due to attribute (concept) incompleteness. Attribute incompleteness is present when identical examples are present in multiple classes (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 and Hieb, 1992). In this approach 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. In addition the questions are guided by the 'gaps' in the knowledge base so they are carefully constructed and useful.

3.3 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 or other significance measures applied to learned hypotheses. Some tree-pruning methods include (Quinlan, 1986) and (Mingers, 1989). Pruning methods applied to learned rules include the AQ family of programs (Michalski, 1986; Zhang, 1989), and CN2 (Clark, 1989) and pruning applied to the training data based on rule-weight is presented in (Pachowicz, Bala and Zhang, 1992).

4. An Empirical Comparison
4.1 Descriptions of methods evaluated

As mentioned earlier a number of methods for constructive induction have been developed. Most of these methods modify the representation space by adding new attributes. Other methods exist which abstract attribute values or remove irrelevant attributes. Although each of these methods has been developed with a specific representation space transformation in mind it is important to empirically measure how effective each of these methods are in achieving this goal and, in the case of competing methods, determining differences. AQ17 is a system which combines a number of different methods for representation space modification. The selection of different methods is performed by the user. All of the methods use the AQ inductive learning module for performing the search for inductive hypotheses. A single method for hypothesis generation is used because the types of rules learned by AQ are comprehensible and powerful and it eases the control of the double search problem by forcing the representation space modifiers to correct the space for a single type of hypothesis language construct.

The AQ17 system includes a number of representation space modifiers. These modifiers are briefly described below:

1) Attribute construction

a) Hypothesis-driven CI (HCI) is a method for constructing new attributes based on an analysis of inductive hypotheses. Useful concepts in the rules can be extracted and used to define new attributes. These new attributes are useful because explicitly express hidden relationships in the data. This method of hypothesis analysis as a means of constructing new attributes is detailed in a number of places including (Wnek, 1993, Wnek and Michalski, 1994).

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

2) Attribute value modification

Attribute-value modification can be either the addition, (concretion) of values to an existing attribute domain, or the deletion (abstraction) of attribute values. Currently the program which performs this modification, SCALE, implements both a χ^2 method and an equal-interval-size method. The χ^2 method calculates the correlation between an attribute-value interval and the class. Using a χ^2 correlation to quantize data was first proposed by Kerber (Kerber, 1992). Attribute value modification (AVM) selects a set $V' \subset V$ (where V is the domain of A) of allowable values for attribute A. AVM can be used to reduce multi-valued nominal domains, or real-valued continuous data into useful discrete, values.

3) Attribute removal

A hypothesis-driven method can also be used to perform attribute removal. Attribute removal makes a selection of a subset X' of attributes from the original attribute set X. In AQ17, a logic-based attribute removal is performed. The irrelevancy of an attribute is calculated by analyzing generated hypotheses. 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.

To empirically determine the effectiveness of these methods for representation space modification they were applied to a set of artificial problems.

4.2 Problem Descriptions

One goal of this report is to empirically determine the types of problems that are best suited to two methods for constructive induction: data-driven and hypothesis-driven. To this end a set of artificial DNF type problems were generated. In each problem there are 500 total instances, 70% are used for training and 30% are used for testing. 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 five different problems are: 1) random incorrect instance labelling (misclassification noise), 2) incorrect attribute-values 3) inappropriate attribute values (overly large attribute domain sizes) 4) inappropriate attributes (irrelevant attributes) and 5) inappropriate attributes (relevant to the target concept, but resulting in an example distribution which is difficult for the given hypothesis language to describe).

The artificial problems consist of a set of six DNF type concepts. A description of each of the six problems and the goal concept for the positive class in each is given below:

1) Problem t0 original DNF
Positive class: [x1= 4,5] [x2=1..3][x3=1,2] v [x3=4,5][x4=2][x5=2]

2) Problem t1 (25% of the training instances misclassified)
Positive class: [x1= 4,5] [x2=1..3][x3=1,2] v [x3=4,5][x4=2][x5=2]

3) Problem t2 (attribute value noise: 187 of the training examples have one or more attributes whose values have been modified).
Positive class: [x1= 4,5] [x2=1..3][x3=1,2] v [x3=4,5][x4=2][x5=2]

4) Problem t3 (inappropriate attribute-value set/overprecision: the domain of all of the attributes has been increased from 6 to 60)
Positive class: [x1= 40..59] [x2=10..39][x3=10..29] v [x3=40..59][x4=20..29][x5=20..29]

5) problem t4 (inappropriate attributes: the decimal value of x3 has been mapped using a 6 place parity coding, e.g. 3 = 001011. The selection of a particular equivalent coding is random)
Positive class: [x1= 4,5] [x2=1..3][[#attributes(x6..x11)=1]=1,2] v [#attributes(x6..x11)=1]=4,5][x4=2][x5=2]

6) Problem t5 (40 irrelevant attributes added)
Positive class: [x1= 4,5] [x2=1..3][x3=1,2] v [x3=4,5][x4=2][x5=2]

4.3 Results

These results show the predictive accuracy of AQ (no representation space modification), AQ-HCI, AQ-HCI(ADD), AQ-HCI(REMOVE), AQ-DCI and AQ-SCALE.

% Correct (strict match)						
Problem	T0	T1	T2	T3	T4	T5
AQ	100	71	83	66	91	86
AQ-HCI	100	76	90	76	91	68
AQ-HCI (ADD)	100	76	90	76	91	87
AQ-HCI (REMOVE)	100	71	83	66	91	68
AQ-DCI	100	65	84	70	100	78
AQ-SCALE	100	70	57	77	94	51

Table 1. Test results of 6 representation space modifiers on 6 artificial problems

These test results reveal some of the characteristics of the six methods for representation space modification. It is helpful in this discussion to analyze pairs of results. The pairs being compared are:
1) Representation space change vs. no change (AQ-HCI, AQ-HCI(ADD), AQ-HCI(REMOVE), AQ-DCI and AQ-SCALE vs. AQ).
2) AQ-DCI attribute construction vs. AQ-HCI attribute construction
3) construction vs. contraction (AQ-HCI(ADD) and AQ-DCI vs. AQ-SCALE and AQ-HCI(REMOVE))

Representation change vs. No change

Invoking representation space modifications to a simple problem (t0) do not adversely affect the ability of AQ to learn the goal concept. Despite the construction of new attributes or the removal of attribute values (AQ-HCI(REMOVE) made no changes to the space) the simple correct rules were found. If the problem is not so clear, the effects of representation space change can be either helpful or destructive.

AQ-HCI vs. AQ

AQ-HCI(ADD) significantly improved the performance over AQ alone for the problems of misclassification (t1), attribute value noise (t2), overprecision (t3) and irrelevant attributes (t5). AQ-HCI(ADD) construction caused no change in accuracy for the distributed coding problem (t4).

HCI removal made changes to only the distributed coding problem (t4) and, as expected the irrelevant attribute problem (t5). In the latter problem it removed 14 of the 45 total attributes. Unfortunately AQ-HCI(REMOVE) also removed two relevant attributes without which the goal concept for the positive class could not be found. This result highlights the importance of understanding how an attribute removal is performed. In AQ-HCI(REMOVE) removal is based on the presence (or lack) of attributes in the generated rules. If an attribute is used by AQ it is considered irrelevant and removed. This removal can occur independent of the *information value* of an attribute: x2, x3 and x6 were retained despite an information value of 0.00 while x44 and x45 with information values of 0.08, and 0.06 respectively were removed. When a threshold-filter using individual information value was used to filter the data AQ was able to learn a set of 7 rules that had a predictive accuracy of 100%.

AQ-HCI achieved the same results as AQ-HCI(ADD) for all but the irrelevant attribute problem (t6). In this problem the negative effects of the removal of two relevant attributes (x44 and x45) was slightly offset by the construction of two new attributes.

AQ-DCI vs. AQ

AQ-DCI attribute construction slightly improved the performance of AQ for problems that had attribute value noise (t2: 84% accuracy and 29 rules vs. 83% accuracy and 33 rules) and overprecision (t3: 70% vs. 66% accuracy). As expected DCI construction significantly improved performance for the distributed coding problem (t4: 100% accuracy and 7 rules vs. 91% accuracy and 26 rules).

AQ-DCI attribute construction resulted in decreased performance of AQ for problems of misclassification (t1: 71% accuracy vs. 65% accuracy each with 53 rules) and irrelevant attributes (t5: 78% accuracy vs. 86% accuracy). AQ-DCI attribute construction appears to build attributes which strongly fit the training data. If this data has is correct and appropriate, then these strong new attributes will help the learner find simple, accurate rules, otherwise the training data will be overfit and poor rules will result.

AQ-SCALE vs. AQ

AQ-SCALE attribute value abstraction resulted in improved performance of AQ for problems of overprecision (t3: 77% accuracy vs. 66% accuracy) and distributed coding (t4: 94% accuracy vs. 91% accuracy). AQ-SCALE attribute value abstraction resulted in decreased performance for misclassification noise (t1: 70% vs. 71%), attribute value noise (t2: 57% accuracy vs. 83% accuracy), and irrelevant attributes present (51% accuracy vs. 86% accuracy). As can be expected for a method that removes information, AQ-SCALE can significantly increase performance for problems that are defined too precisely, but it can also significantly reduce performance by removing relevant attribute values.

AQ-HCI attribute construction vs. AQ-DCI attribute construction

AQ-HCI(ADD) attribute construction performed better than AQ-DCI attribute construction on all problems except the distributed coding problem (t4: AQ-DCI 100% accuracy and 7 rules vs. AQ-HCI(ADD) 91% accuracy and 12 rules). This suggests that AQ-DCI is capable of correcting specific inadequacies in the data (and significantly increasing the accuracy of resulting rules) , but that AQ-HCI(ADD) is a more general tool for improving the representation space and does not cause a significant decrease in performance. There is tradeoff here between power and simplicity in

the DCI approach and the generality, but complexity of the resulting AQ-HCI(ADD) generated attributes.

Representation space expansion vs. representation space contraction

Not surprisingly the construction of new attributes is a safer modification to the representation space than the contraction of that space through attribute value abstraction (AQ-SCALE) and attribute removal (AQ-HCI(REMOVE)). The average performance of the two attribute construction methods over all 6 problems is 84.75% while the average for the destruction methods is 77.33%.

5. Conclusion

In this paper four different methods for representation space modification were applied to six different artificial problems. The methods performed representation space expansion (AQ-DCI, and AQ-HCI(ADD)) and contraction (AQ-SCALE, AQ-HCI(REMOVE)). The results highlighted the strengths and weaknesses of the different approaches. Expansion methods are safer than contraction methods, and can significantly increase performance when such methods are well tuned to the problem at hand. Domain knowledge can be very useful in achieving this. Although only one example from each was tested, it appears data-driven methods are very sensitive to irrelevant attributes and misclassification noise and therefore should be avoided when such problems are known to be present. In addition, hypothesis-driven construction was found to be generally useful, although it achieved this at the sacrifice of some simplicity. Contraction methods can result in significant reduction in performance accuracy and should be used very carefully.

References

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., Michalski, R.S. and Wnek, J., "Multistrategy Constructive Induction: AQ17-MCI," *Second International Workshop on Multistrategy Learning*, pp. 188-203, Harpers Ferry, WV, 1993.

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

Clark, P., and Niblett, T., "The CN2 Induction Algorithm," *Machine Learning*, Vol. pp. 1989.

Drastal, G., Czako, G. and Raatz, S., "Induction in an Abstraction Space: A Form of Constructive Induction," *Proceedings of IJCAI-89*, pp. 707-712, Detroit, MI, 1989.

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

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

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

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

Palo Alto, CA: Morgan Kaufmann, 1983.

Mingers, J., "An Empirical Comparison of Pruning Methods for Decision-Tree Induction," *Machine Learning,* 2, Vol. pp. 1989.

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

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

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

Quinlan, J.R., *The Effect of Noise on Concept Learning*, Morgan Kaufmann, Los Altos, CA, 1986.

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," *Tenth International Workshop on Machine Learning*, pp. 66-78, 1993.

Tecuci, G., and Kodratoff, Y., *Apprenticeship Learning in Imperfect Domain*, Morgan Kaufmann, Palo Alto, 1990.

Tecuci, G., and Hieb, M., "Consistency-driven Knowledge Elicitation: Using a Learning oriented Knowledge Representation that Supports Knowledge Elicitation in NeoDISCIPLE," Machine Learning and Inference Laboratory, GMU, MLI 92-8, 1992.

Utgoff, P.E., "Shift of Bias for Inductive Learning," *Machine Learning: An Artificial Intelligence Approach, Vol. II*, J.G. Carbonell and T.M. Mitchell (Eds.) R.S. Michalski (Eds.), Morgan Kaufmann, Los Altos, CA, 1986.

Wnek, J., *Hypothesis-driven Constructive Induction*, George Mason University, PhD, 1993.

Wnek, J., and Michalski, R.S., "Discovering Representation Space Transformations for Learning Concept Descriptions Containing DNF and M-of-N Rules," *Working Notes of the ML-COLT94 Workshop on Constructive Induction*, pp. New Brunswick, NJ, 1994.

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

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

Zhang, J., and Michalski, R.S., "A Preference Criterion in Constructive Learning: A Discussion of Basic Issues," *Proceedings of the 6th International Workshop on Machine Learning*, pp. 17-19, Ithaca, NY, 1989.