Proceedings of the First International Workshop on Intelligent Adaptive Systems (IAS-95)
Ibrahim F. Imam and Janusz Wnek (Eds.), pp. 38-51, Melbourne Beach, Florida, 1995.

Constructive Induction-based Learning Agents: An Architecture and Preliminary Experiments

Eric Bloedorn and Janusz Wnek
Center for Machine Learning and Inference
George Mason University
4400 University Dr., Fairfax VA 22030, USA
{bloedorn, wnek}@aic.gmu.edu

Abstract

This paper introduces a new type of intelligent agent called a constructive induction-based learning agent (CILA). This agent differs from other adaptive agents because it has the ability to not only learn how to assist a user in some task, but also to incrementally adapt its knowledge representation space to better fit the given learning task. The agent's ability to autonomously make problem-oriented modifications to the originally given representation space is due to its constructive induction (CI) learning method. Selective induction (SI) learning methods, and agents based on these methods, rely on a good representation space. A good representation space has no misclassification noise, inter-correlated attributes or irrelevant attributes. Our proposed CILA has methods for overcoming all of these problems. In agent domains with poor representations, the CI-based learning agent will learn more accurate rules and be more useful than an SI-based learning agent. This paper gives an architecture for a CI-based learning agent and gives an empirical comparison of a CI and SI for a set of six abstract domains involving DNF-type (disjunctive normal form) descriptions.

Key words: intelligent agents, constructive induction, multistrategy learning.

1. Introduction

The goal of research in intelligent agents is to construct software that can provide individualized assistance to users. Two approaches that have been used in the past are 1) to force the end-user to provide the necessary skills by programming the agent, or 2) to provide the agent with a priori domain-specific knowledge about the application and user. The first approach is too difficult for most users, and the second approach is too hard for application developers, who must accurately predict the current and future needs of users (Maes, 1994).

Another proposed approach is to build into the agents an ability to learn required skills from experience (Dent, 1992, Maes, 1994). In this method, the agent gains competence by interacting

with the user as the user performs some tasks. This agent learns in four different ways: 1) by observing the user, 2) from user's feedback, 3) from user provided training examples, and 4) by interaction with other agents. The ability to modify the representation space is an important element in all four types of learning.

Previously reported research in building learning agents uses a pre-defined set of attributes to describe the learning example. For example, in describing e-mail messages, the Maxims agent (Maes, 1994) uses features such as the sender and receiver of the message and key words in the "Subject" field. CAP, an agent for meeting scheduling (Mitchell, 1994) uses features such as event_type, time and duration. CAP does automatically calculate the values of a number of features such as number-of-attendees, single-attendee, but the feature set is still pre-defined.

The ability of an agent to learn useful, individualized skills is, like any learning task, strongly affected by the representation of the problem. For example, suppose student Blake is a user which prioritizes his e-mail messages according to the number of people receiving the message. i.e. a message with 1 receiver (Blake) is prioritized as important, while thirty-five receivers (his class), is prioritized lower. An e-mail agent without the ability to extract “number of receivers” from the FROM line will not quickly detect this simple preference. This failure will result in Blake’s agent being of little value to him. An agent which can automatically extend its set of features, in this case by adding a feature which counts the number of e-mail recipients, will be able to overcome this limitation and be of great use to Blake. Other simple modifications may include “Number-of-messages-from-USER”, or “Number-of-messages-about-SUBJECT”.

The ability to simultaneously search for an ‘adequate’ representation space, and for a hypothesis within that space is known as Constructive Induction (CI) (Figure 1). Agents which have the ability to automatically modify their representation space of their given learning task using CI are known as Constructive Induction-based Learning Agents (CILA). This ability allows these agents to overcome representation space problems such as misclassification noise, inter-correlated attributes and irrelevant attributes. Given a representation space better suited to learning these agents can more quickly adapt to the user needs.

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).

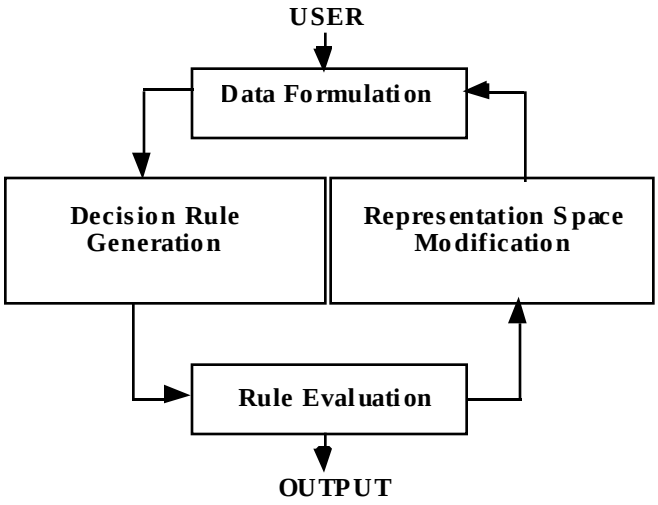

Figure 1. Constructive Induction viewed as a search for both the best representation space and for the best hypothesis (decision rules).

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 AQ-type learning systems use methods such as "dropping conditions," "extension against," "adding an alternative," "closing interval," and "climbing a generalization tree." The representation space search operators 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.

2. The Need for Representation Space Modifiers

Methods for building intelligent agents which use machine learning based on selective induction will have all the weaknesses of this strategy. These weaknesses become increasingly important as attempts are made to move machine learning methods into real-world applications such as scheduling meetings and e-mail filtering. This section describes some of the weaknesses of selective induction methods and describes work in constructive induction aimed at overcoming these problems.

As mentioned earlier, constructive induction divides the process of creating new knowledge into a phase that determines the "best" representation, and into a phase that actually formulates the "best" decision rules. The reason for such a division is that original representation space is often inadequate for representing a given learning task. To illustrate this problem, consider Figure 2a. Let us suppose that the problem is to construct a general description that separates points marked

by + from points marked by -. In this case, the problem is easy, because "+" points are clearly separated in the representation space from "-" points. One can place all "+"s in a rectangle, or draw a straight line between "+"s and "-"s. Let us suppose now that we have a similar problem, but the "+"s and "-"s are distributed as in Figure 2b. In this case, it is not easy to separate the two groups. This is an indication that perhaps the representation space is inadequate for the problem at hand.

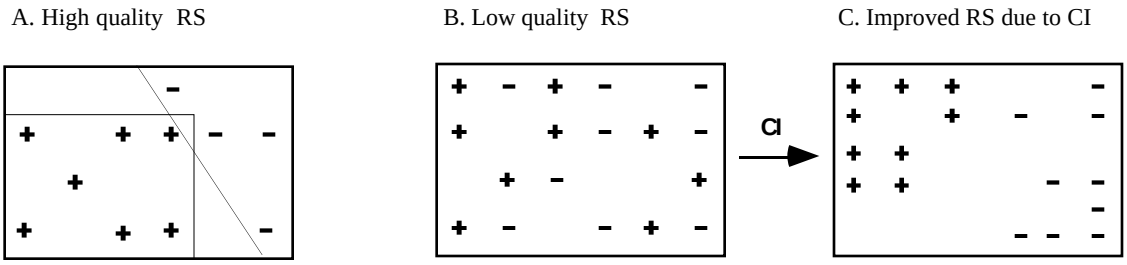

Figure 2. High versus low quality representation spaces for concept learning.

A traditional approach, implemented in selective induction systems, is to draw complex boundaries that will separate these two groups. The constructive induction approach is to search for a better representation space (Figure 2c), in which the two groups are well separated. Conducting constructive induction thus requires mechanisms for generating new, more problem-relevant dimensions of the space (attributes or descriptive terms), as well as modifying or removing less relevant dimensions from among those initially provided. Therefore, a constructive induction system performs a problem-oriented transformation of the knowledge representation space. Once an appropriate representation space is found, a relatively simple learning method may suffice to develop a desirable knowledge structure (in this case, a description that separates the two groups of points).

The type of hypotheses languages used can affect which problems are difficult and which are easy. This is sometimes known as inductive bias. Some problems, like the one represented in Figure 2b, however, are difficult for any set of hypothesis constructs. We categorize the source of this representational difficulty into two classes: incorrectness or inappropriateness. 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.

Problem incorrectness occurs when some attribute-values, attributes or instances are incorrectly labeled. This may occur if the user mistakenly labels an example with an unintended class. Incorrectness is most often associated with noise in the training data due to inconsistency in the acquisition of examples. 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).

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 examples 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. 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. Overprecision 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 systems 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. An Architecture for a Representation Space Adapting Agent

In order to build an intelligent agent that can gain enough competence to be useful to an individual that agent must acquire a great deal of knowledge. A method which uses machine learning to automatically acquire this knowledge is only as successful as the learning technique being used. Selective induction based intelligent agents will fail when the provided representation space is inadequate for the learning task. An constructive induction-based learning agent is able to expand or contract the provided representation space either automatically, or based on knowledge provided by the user using one or more of the different types of CI: data-driven, hypothesis-driven, knowledge-driven and multistrategy (Wnek and Michalski, 1994).

An architecture for a constructive induction-based learning agent is shown in Figure 3. In this architecture the agent acts as an assistant to the user in dealing with the environment. The user can access the environment directly or through the agent. In its passive monitor mode the agent records the actions of the user when the environment is accessed directly. This record allows the agent to improve its performance without active involvement of the user. In the active mode the agent learns the skills it needs to be useful to the user (such as the user's preferences in reading news articles) based on an iterative interaction with the user. The agent's current understanding of the problem, such as a user profile and the domains of known features are stored in the knowledge base. The contents of the knowledge base are updated by the constructive induction learning algorithm.

The learning component of this agent uses constructive induction so both the profiles and the representation space in which these profiles are described is modified. The representation space modification module may 1) add a new feature, 2) remove a feature, 3) add a new feature value through interaction with the user, or 4) remove a feature value. Adding or removing features may be based on data-driven CI (DCI), hypothesis-driven CI (HCI) or knowledge-driven CI (KCI). Removing feature values is possible via DCI, HCI or KCI. Adding feature values is currently only possible via KCI. In knowledge-driven constructive induction the user provides direct guidance on how to modify the representation space. In data-driven CI, the agent automatically generates modifications based on an analysis of the data, i.e., the examples acquired from observing the user. In hypothesis-driven CI, the agent modifies the representation space based on an analysis of the learned, or provided hypotheses. Using this approach Blake's need for a new attribute could be detected, and corrected in many different ways.

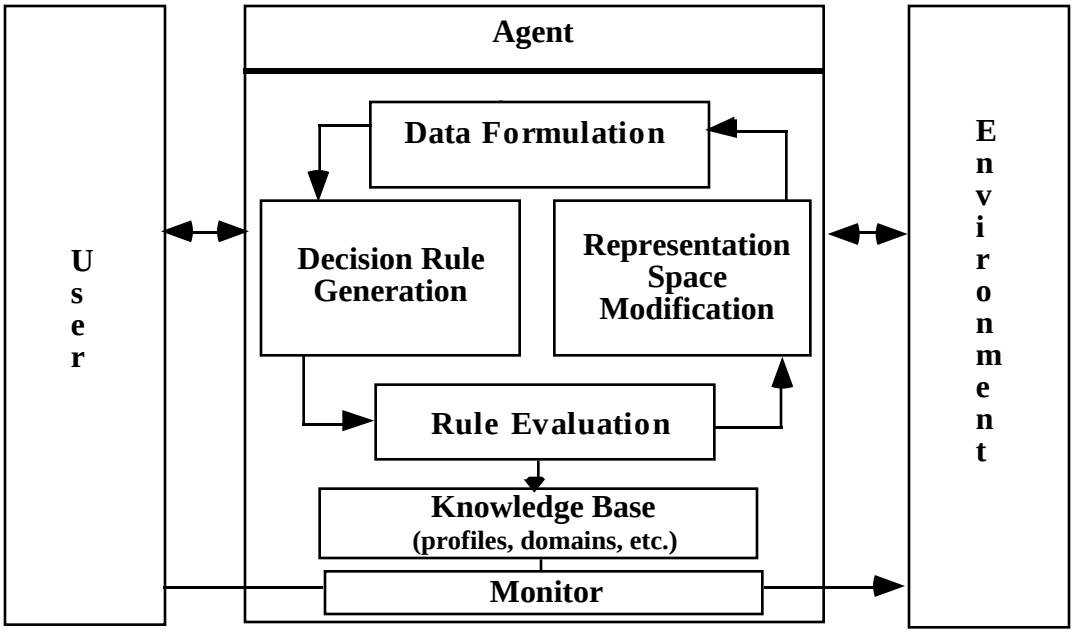

Figure 3. An architecture for a constructive induction-based learning agent.

The specific algorithms used in the representation space modification module are not detailed in this architecture. This architecture can thus describe a wide variety of approaches to building CI-based learning agents. The most appropriate set of modification operators is difficult to determine generally. The next section tries to find an answer to that problem. Section 4 describes an experiment in which a set of six different representation space modification (RSM) operators are each applied to a set of six problems. Rules are learned after each RSM application. These results are compared to the learned rules without any modification. The results show the superiority of a system which combines these six RSM operators over a system with only one operator, or without any automated representation space modification.

4. An Empirical Comparison

4.1 Descriptions of methods evaluated

In order to determine the effectiveness of different CI methods, a set of experiments was performed. These experiments are described in greater detail in (Bloedorn, et. al, 1994). This set of experiments samples a wide variety of possible learning problems including: misclassification noise, attribute-value noise, overprecision, inappropriate attributes and irrelevant attributes. In all of these experiments the AQ15c program was used as the learning algorithm (Wnek et al., 1995). Each of the CI methods must transform the difficult problem into one in which AQ15c would learn simple, predictively accurate rules. A single method for hypothesis generation is used because the types of rules learned by AQ are comprehensible and efficient in decision making. The modifiers

compared in these experiments 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.

The effectiveness of the combination of each of these representation space methods versus a

selective induction method was determined based on a set experiments.

4.2 Problem Descriptions

To test the usefulness of the available methods for representation space modification a set of abstract DNF—type problems were generated. The problems were generated so that the types of problems tested could be carefully controlled. 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 modifications to the original problem are: 1) random incorrect instance labeling (misclassification noise), 2) incorrect attribute-values 3) inappropriate attribute values (overly large attribute domain sizes) 4) inappropriate attributes (attributes relevant to the target concept but causing difficult to describe distribution of examples), and 5) irrelevant attributes. A description of each of the six problems and the goal concept for the positive class in each is given below:

T0 Original DNF
Positive class: [x1= 4,5] & [x2=1..3] & [x3=1,2] v [x3=4,5] & [x4=2] & [x5=2]

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]

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]

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]

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]

T5 (40 irrelevant attributes added)
Positive class: [x1= 4,5] & [x2=1..3] & [x3=1,2] v [x3=4,5] & [x4=2] & [x5=2]

5. Results

Table 1 shows the prediction accuracy of rules learned from examples by the selective induction system AQ15c (with no representation space modification), and after each of the six representation space modification operators were applied.

Prediction accuracy on selected problems of a variety of CI methods						
Method	**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. Prediction accuracies of learned rules on six variations on a DNF—type problems.

The results of these experiments show that no single RSM method is best for solving the wide variety of difficulties possible. These problems include (t1) misclassification noise, (t2) attribute-value noise, (t3) inappropriate attribute-value precision, (t4) inappropriate attributes and (t5) irrelevant attributes. A simple method for combining the strengths of each of the individual CI methods is to run them all separately, and select the best based on results of a secondary testing set. The results shown in Table 2 are based on a SILA agent using the selective induction learning module AQ15c, and CILA agent using AQ15c equipped with the best of the six RSM operators.

SILA vs. CILA prediction accuracy on selected problems						
Method	**T0**	**T1**	**T2**	**T3**	**T4**	**T5**
SILA	100	71	83	66	91	86
CILA	100	76	90	77	100	87

Table 2. Predictive accuracies of a selective induction-based agent versus a constructive induction-based learning agent using a simple combination architecture.

6. Conclusion and Future Work

This paper introduced a novel general architecture for an intelligent agent that allowed this agent to learn from experience, but not be bound by the original representation of the problem. This constructive induction-based learning agent (CILA) is capable of modifying the representation

space. The results of a comparison between an SI learning method and a CI learning method (which includes multiple methods for representation space modification) show that a CI-based learning agent will be more robust to a variety of representation problems. As intelligent agents are applied to problems outside careful controls and by non-experts, the ability of these agents to be robust is increasingly important.

This preliminary work suggests that an agent will need to have some form of constructive induction in order to overcome the difficulties that exist in real-world learning situations. However, because no single method for constructive induction performed best for all the problems posed an approach for combining methods is needed. Further work which details the area of applicability of individual constructive induction methods is needed. Preliminary work in this area is described in (Bloedorn et. al, 1993).

Acknowledgements

This research was conducted in the Center for Machine Learning and Inference at George Mason University. The Center's research is supported in part by the Advanced Research Projects Agency under 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, in part by the Office of Naval Research under Grant No. N00014-91-J-1351, and in part by the National Science Foundation under Grants No. IRI-9020266, CDA-9309725 and DMI-9496192.

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.

Bloedorn, E., Michalski, R.S. and Wnek, J., "Matching Methods with Problems: A Comparative Analysis of Constructive Induction Approaches," *Reports of the Machine Learning and Inference Laboratory*, MLI 94-2, Center for AI, George Mason University, Fairfax, VA, 1994.

Clark, P. and Niblett, T., "The CN2 Induction Algorithm," *Machine Learning*, Vol. 3, pp. 261-284, 1989.

Dent, L., Boticario, J., McDermott, J., Mitchell, T. and Zabowski, D., "A Personal Learning Apprentice," *Proceedings of Tenth National Conference on AI*, San Jose, CA, July 12-16 1992.

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.

Maes, P., "Agents that Reduce Work and Information Overload," *Communications of the ACM*, Vol. 37, No. 7, pp. 31-40, 1994.

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*, Vol. 2, 1989.

Mitchell, T., Caruana, R., Freitag, D., McDermott, J. and Zabowski, D., "Experience with a Personal Learning Assistant," *Communications of the ACM*, Vol 37, No. 7, pp. 81-91, 1994.

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. 5, pp. 71-99, 1990.

Quinlan, J.R., *C4.5: Programs for Machine Learning*, San Mateo, Morgan Kaufmann, CA, 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," *Computational Intelligence*, Vol. 6, pp. 247-270, 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.

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*, PhD dissertation, School of Information Technology and Engineering, George Mason University, Fairfax, VA, University Microfilms International, Ann Arbor, MI, 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*, 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. 14, pp. 139-168, 1994.

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

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.