Adding Domain Knowledge to SBL through Feature Construction

Christopher John Matheus*
GTE Laboratories Incorporated
40 Sylvan Rd., Waltham MA 02254
cjm1@bunny.gte.com

Abstract

This paper presents two methods for adding domain knowledge to similarity-based learning through *feature construction*, a form of representation change in which new features are constructed from relationships detected among existing features. In the first method, domain-knowledge constraints are used to eliminate less desirable new features before they are constructed. In the second method, domain-dependent transformations generalize new features in ways meaningful to the current problem. These two uses of domain knowledge are illustrated in CITRE where they are shown to improve hypothesis accuracy and conciseness on a tic-tac-toe classification problem.

Introduction

One advantage of explanation-based learning (EBL) is its ability to learn from few examples by exploiting domain-specific constraints represented in a domain theory. Similarity-based learning (SBL), on the other hand, requires relatively large numbers of training instances, but is more readily applicable because a domain theory need not be available. Recent research in machine learning has begun to focus on methods of integrating EBL and SBL techniques (see the section entitled *Combining Empirical and Explanation-based Learning* in [Segre, 1989]). This paper proposes an integrated approach that incorporates domain knowledge into SBL systems through feature construction.

Feature construction is the process of defining new features based on useful relationships discovered among existing features. Constructed features are typically used to re-represent the training instances in hopes of making the target concept easier to learn [Matheus, 1989]. Because the space of new features is usually intractable to search, practical feature construction requires strong and appropriate constraints. Domain knowledge can provide the appropriate constraints, similar to the way that domain theories constrain learning in EBL. Unlike EBL, however, domain knowledge used in feature construction need not be a complete theory, but may comprise simple, disjoint pieces of problem-specific information. The use of this sort of knowledge-driven feature construction provides a way of adding simple domain knowledge to SBL systems. Since only the features representing the training instances are affected by this approach, the underlying inductive algorithm need not be altered.

Several machine learning systems perform feature construction; recent examples include DUCE [Muggleton, 1987], FRINGE [Pagallo, 1989], MIRO [Drastal and Raatz, 1989], PLS0 [Rendell, 1985], STABB [Utgoff, 1986], and STAGGER [Schlimmer, 1987] (see [Matheus, 1989] for a description of these and other feature construction systems). Only a few of these systems, however, explicitly use domain knowledge during feature construction (*e.g.*, MIRO). This paper describes two methods for using domain knowledge in feature construction, and outlines their implementations in CITRE [Matheus and Rendell, 1989, Matheus, 1989]. Experimental results are presented that demonstrate the successful application of these methods on a tic-tac-toe classification problem.

CITRE

CITRE is a decision-tree-based learning system that performs feature construction by selecting relationships for new features from positive tree branches. Although similar in this respect to FRINGE [Pagallo, 1989], CITRE differs in its use of a variety of new-feature selection methods, its use of domain knowledge to filter out undesirable features, its potential for generalizing new features, and its evaluation of constructed features.

Figure 1 illustrates CITRE's learning algorithm. A learning problem is submitted to CITRE as a set of original features $F = \{p_1, ..., p_m\}$ called *primitives* and a set of training instances $I = \{i_1, ..., i_n\}$ described in terms of the primitives. The instances and features are used to construct an initial decision tree based on the information-theoretic splitting criteria employed by ID3 [Quinlan, 1983]. The constructed tree is then

*This research was supported by a University of Illinois CS/AI Fellowship, the National Science Foundation, Grant No. IRI 8822031, and the Office of Naval Research, Grant No. N00014-88K-0124.

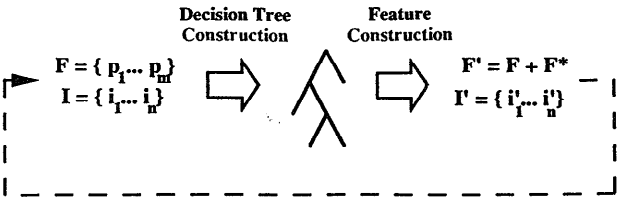

Figure 1: Illustration of CITRE's learning algorithm.

pruned using pessimistic pruning [Quinlan, 1987], and passed as input to CITRE's feature-construction module. New features are constructed by selecting relationships from the decision tree as described below. The new features F^* are added to the active feature set, and the entire process is repeated. This iterative learning algorithm terminates when either the current tree contains no positive branches consisting of more than one feature, or all potential new features from the current tree are already present in the active feature set. Although the version of CITRE described in this paper constructs only Boolean features, the system is capable of working with learning problems having nominal, ordinal, or continuous-valued primitives.

Feature Construction in CITRE

Feature construction can be viewed in terms of four aspects (see [Matheus, 1989]):

1. the *detection* of when new features are required
2. the *selection* of relationships used to define new features
3. the *generalization* of new features
4. the global *evaluation* of constructed features

Descriptions of CITRE's approach to detection, selection, and evaluation can be found in [Matheus and Rendell, 1989, Matheus, 1989]. This paper focuses on CITRE's methods of domain-knowledge filtering during selection and new-feature generalization.

During *selection*, CITRE selects pairs of Boolean relations (*i.e.*, feature-value pairs) from the nodes in positively labeled branches of a decision tree, and conjoins them to form new Boolean features, *e.g.*, *and*(*color* = *red*, *size* = *big*). This selection is achieved by one of five methods: *root*, *fringe*, *root-fringe*, *adjacent*, or *all*. The *root* method selects the relations in the first two nodes of each positive branch, the *fringe* method selects the last two (*i.e.*, the method used by FRINGE), the *root-fringe* method selects both root and fringe pairs, the *adjacent* method selects all adjacent pairs along each branch, and the *all* method selects every possible combination of feature pairs along each individual branch. The *adjacent* selection method was used for the experiments described in this paper (see [Matheus, 1989] for comparisons of other methods).

The selected pairs of relations are passed through a *domain-knowledge filter* eliminating pairs that do not satisfy the constraints imposed by the available domain knowledge. Domain knowledge is represented in CITRE as predicates that define relationships required of all selected pairs. For example, in the tic-tac-toe experiments described below, information about *piece adjacency* was encoded as a domain-knowledge constraint restricting operands to pairs of spatially adjacent features. A virtually unlimited number of domain-knowledge constraints may be defined in this way. On the other hand, domain-knowledge constraints are not required; in their absence, all selected relation pairs are used to define new features.

Domain-knowledge filtering reduces the space of new features and can result in a more tractable space containing a higher percentage of useful features. Unlike a domain theory in EBL, CITRE's constraints need not define complete relationships between the goal concept and the primitives. As a result, simple commonsense types of information can be made available to the learning algorithm, *e.g.*, in board games spatial proximity is often an important factor.

After selection, CITRE performs *feature generalization* using domain-dependent generalization operators. The available generalization operators are applied sequentially to each potential new feature, resulting in one or more generalized new features. In the tic-tac-toe experiments, four generalization operators were used to spatially translate candidate features up, down, left, and right on the game board. This process of feature generalization can help identify potentially useful features more quickly and with fewer training examples. As with filtering, all available domain knowledge is applied to every new feature.

Together, domain-knowledge filtering and generalization operators provide a simple, systematic method for incorporating specific domain knowledge into feature construction, and thereby into the SBL inductive processes. As the experiments in the next section demonstrate, the addition of domain knowledge in this way can result in significant improvements in hypothesis accuracy, hypothesis conciseness, and overall learning time.

Tic-tac-toe Classification

The use of domain knowledge in CITRE is demonstrated in this section on a tic-tac-toe classification problem. This learning problem was selected because of the disjunctive form of the target concept (which poses difficulties for many SBL algorithms), and because of the availability of simple domain knowledge relevant to board games. The target concept for this classification problem is "a win for x." The feature set comprises nine nominal primitives corresponding to the nine board locations (labeled p11, p12, p13, p21, p22, p23, p31, p32, and p33 in Figure 2). All features range over the values of x, o, and *blank*. Using these

p11	p12	p13
p21	p22	p23
p31	p32	p33

win for x			win for o			draw		
o		x	o	o	o	o	o	x
	x	o		x		x	x	o
x	x	o	x	x		o	x	x
(626)			(316)			(16)		

Figure 2: The tic-tac-toe board on the left shows the feature labels for each of the nine board squares. Examples of win for x, win for o, and draw are shown to the right. Below each example is the number of instances of that type found in the set of all obtainable, final tic-tac-toe boards.

primitives, the target concept can be expressed by the following disjunctive relationship: $[(p11 = x) \wedge (p12 = x) \wedge (p13 = x)] \vee [(p21 = x) \wedge (p22 = x) \wedge (p23 = x)] \vee [(p31 = x) \wedge (p32 = x) \wedge (p33 = x)] \vee [(p11 = x) \wedge (p21 = x) \wedge (p31 = x)] \vee [(p12 = x) \wedge (p22 = x) \wedge (p32 = x)] \vee [(p13 = x) \wedge (p23 = x) \wedge (p33 = x)] \vee [(p11 = x) \wedge (p22 = x) \wedge (p33 = x)] \vee [(p13 = x) \wedge (p22 = x) \wedge (p31 = x)]$.

Although the instance space defined by the nine features has cardinality $3^9 = 19,683$, the rules for tic-tac-toe permit only 958 final tic-tac-toe boards, of which 626 are wins for x, 316 are wins for o, and 16 are draws (see Figure 2). This entire set of instances was used as the test set in all experiments. Ten training sets for each of five training sizes (100, 200, 300, 400, and 500 examples) were randomly drawn from this set.

For each independent variable tested in the experiments below, ten test runs were made at each of the five training sizes while the following dependent variables were recorded: 1) accuracy of the final hypothesis on classifying the test-set instances, 2) length of the final hypothesis measured as the number of primitives required to define all internal nodes, and 3) CPU time consumed during the entire learning process. These recorded values were averaged and statistically analyzed using a t-test with 95% confidence intervals.

CITRE without Domain Knowledge

CITRE was run without domain-knowledge filtering or generalization in this first series of experiments. The results are shown in Figure 3 for the *adjacent* method and for the decision-tree algorithm without feature construction (referred to as the *none* method). Feature construction improves accuracy by greater than 15% at the higher training sizes. The final hypotheses, however, are less concise. Although part of this increase in length is due to the additional nodes required to improve hypothesis accuracy, the *adjacent* method produces excessively long final hypotheses at the higher training sizes (for example, at training size 400, final hypotheses are at least twice as long as necessary for the corresponding level of accuracy). CPU times are also much greater than those for the *none*

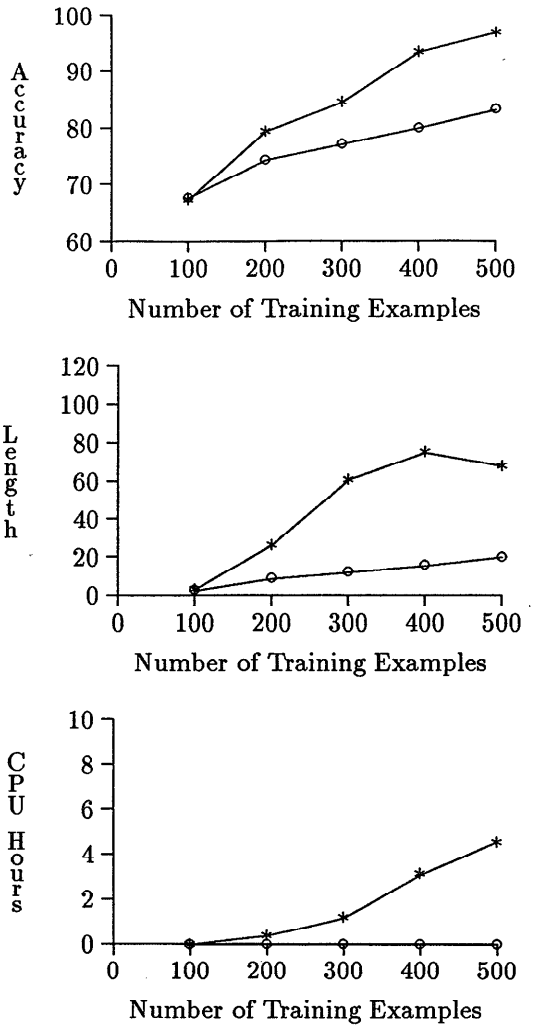

Figure 3: Results of CITRE's application to tic-tac-toe classification without the use of domain knowledge: $* = adjacent$, $\circ = none$.

method (over 700 times as much at training size 500). This increase is due to the additional time required to construct new features and to re-run the tree-induction procedure over several generations. In summary, the *adjacent* method alone improves accuracy significantly but results in less concise hypotheses and requires large amounts of CPU time.

Domain-Knowledge Filtering

The next series of experiments tested the use of domain-knowledge filtering by adding two knowledge constraints: *piece adjacency* and *piece type*. Piece adjacency encodes the importance of piece proximity in board games by constraining new features to relations between features physically adjacent on the game board (*e.g.*, $p11$ and $p12$, but not $p11$ and $p13$). Piece-

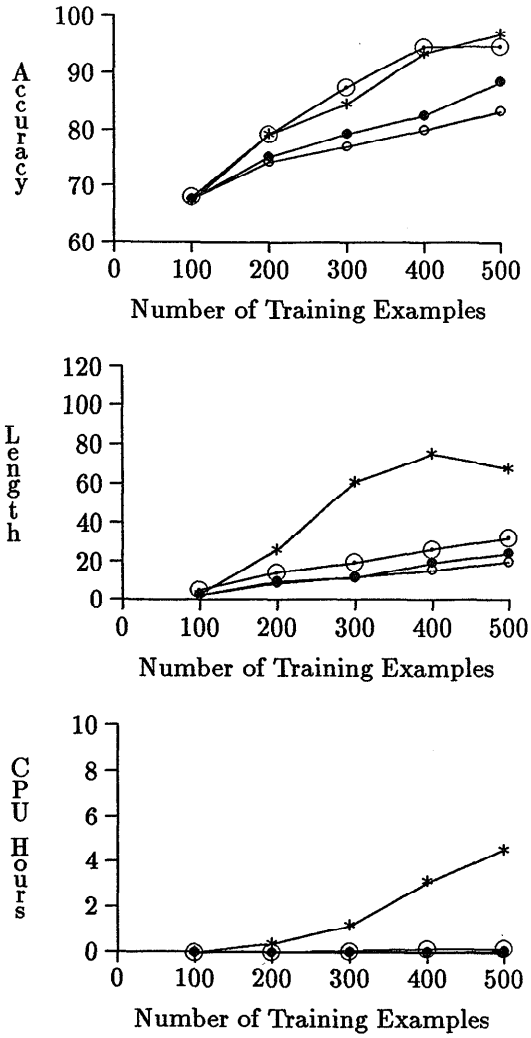

Figure 4: Results with the addition of domain-knowledge filtering: • = *adjacent* + filtering, ⊙ = *adjacent* + filtering with deferred pruning, * = *adjacent*, o = *none*.

type knowledge encodes the importance of distinguishing between the players of the game (x versus o) by constraining new features to relations between features possessing the same value (*e.g.*, $p11 = x$ and $p12 = x$, but not $p11 = x$ and $p12 = o$). New features were required to satisfy both domain-knowledge constraints. Although these pieces of domain knowledge are relevant to a larger class of board-game problems, the specific predicates that defined them in these experiments were tailored for tic-tac-toe.

Figure 4 compares the domain-knowledge filtering results to the previous experiment. Feature construction with filtering improves hypothesis accuracy (significant at size 500), however, the improvement is much less than was achieved with the *adjacent* method without filtering. Although the hypothesis lengths and CPU times are substantially reduced, these improvements are less meaningful in light of the lower accuracy. This poorer performance suggests that the constraints imposed by the domain knowledge were perhaps too restrictive, preventing the construction of useful features. However, although some potentially useful features would have been filtered out for failure to satisfy the piece-adjacency and piece-type constraints, the most useful features were still realizable (*e.g.*, $and(p11 = x, and(p22 = x, p33 = x))$). The reason more of these features were not constructed is that the pruning of trees during feature construction severely reduces the length of positive branches, and thereby the pool of potential new feature (the effects of pruning are discussed more fully in [Matheus, 1989]).

To overcome this problem, an approach was implemented in which pruning is *deferred* until after the final tree is constructed. With this approach feature construction operates on all the information in the unpruned trees, while the benefits of pruning are maintained in the final hypotheses. The results of deferred pruning tests are graphed as the ⊙ plots in Figure 4. For filtering, deferred pruning greatly improves accuracies and has insignificant effects on conciseness and learning time. These results compare favorably with the accuracy results of the pure *adjacent* method (*i.e.*, the * plot in Figure 4).

Feature Generalization

In the next series of experiments new features were generalized by spatially translating them up, down, left, and right on the game board. All translations were made in a single direction but could be extended up to two places in distance. As a result, the minimum number of new features generalizable from a single new feature was zero and the maximum was three. Figure 5 shows the new feature $and(p11 = x, p21 = x)$ translated one place down, one place to the right, and two places to the right, resulting in three new features: $and(p21 = x, p31 = x)$, $and(p12 = x, p22 = x)$, and

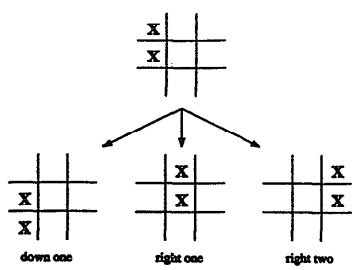

Figure 5: An example of the spatial translations used in feature generalization. The constructed feature $and(p11 = x, p21 = x)$ is translated down one square, right one square, and right two squares to produce three new features.

$and(p13 = x, p23 = x)$. Although the predicates used to perform the generalizations are specific to tic-tac-toe, spatial translations of this sort are relevant to a wide range of learning problems.

As shown in Figure 6, generalization improves accuracy by more than 18% relative to the control method (except at training size 100), and by as much as 10% over the adjacent method alone. Conciseness is not significantly affected by generalization, as the final hypotheses continue to be longer than necessary. CPU times, however, increase to more than 1000 times that required by the control method at size 500. The improved accuracy achieved with generalization would be attractive if conciseness and CPU times could be improved. This situation suggests a combined approach using generalization and domain-knowledge filtering.

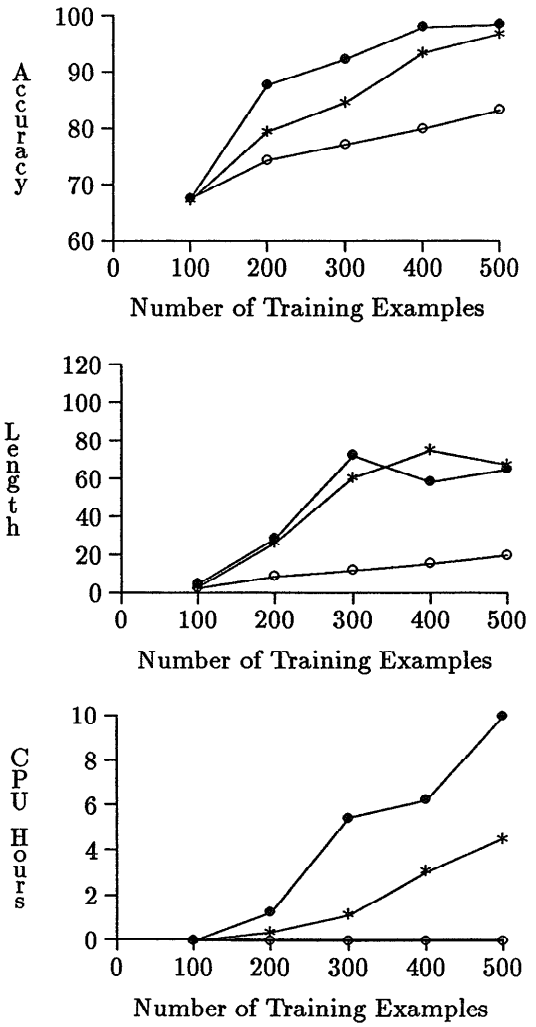

Figure 6: Results with feature generalization: • = *adjacent* + generalization, ∗ = *adjacent*, ○ = *none*.

Combining Filtering and Generalization

The final experiment combined domain-knowledge filtering and feature generalization. The results are graphed in Figure 7 along with plots for the *adjacent* method without filtering or generalization and the *none* method. The combined method significantly improved hypothesis accuracy relative to the control method. Although the accuracies are slightly less than those achieved by generalization alone (except at training size 100), they are slightly better than achieved with filtering alone. As hoped for, the combined method achieves these improved accuracies while reducing hypothesis length and learning times relative to the individual methods of filtering and generalization.

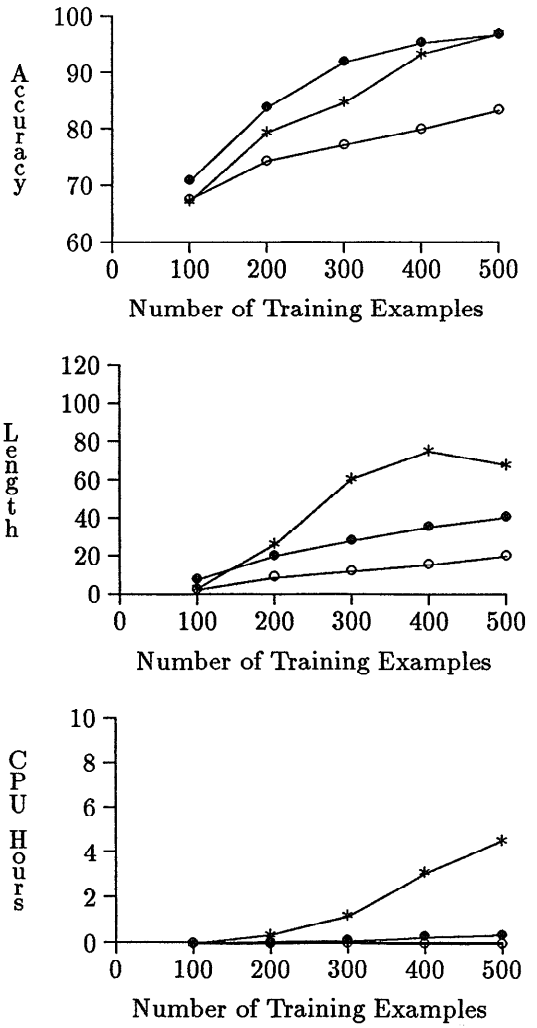

Figure 7: Results for the combined use of generalization and domain-knowledge filtering: • = *adjacent* + generalization + filtering (with deferred pruning), ∗ = *adjacent*, ○ = *none*.

Conclusions

An approach was presented for incorporating domain knowledge into SBL through feature construction. Its use was demonstrated in a version of CITRE that incorporates domain knowledge during feature construction in two ways: as constraints on the types of features that may be constructed, and as transformations for generalizing new features. In empirical tests on tic-tac-toe classification, domain knowledge used in this way improved hypothesis accuracy and conciseness, and reduced the computational costs of feature construction.

This approach offers two main advantages. First, because the domain knowledge is used in feature construction, the underlying inductive algorithm is not affected. As a result, this approach should be readily applicable to existing SBL systems through the addition of a feature construction component. Second, this approach works with domain knowledge ranging in quality from irrelevant information to complete domain theories. When the domain knowledge is inappropriate, few if any useful features will be constructed and performance will fall to the level achievable using the primitives alone. With complete domain knowledge, useful features can be constructed from few examples and optimal accuracy and conciseness can be achieved.

The specific results of CITRE's application to tic-tac-toe classification were presented to demonstrate the central idea of using domain knowledge during feature construction. There are many improvements that could be made both to the general approach and to the specific techniques used in CITRE. In particular, the use of domain-knowledge filtering as described in this paper is rather rigid: if a new feature is inconsistent with the knowledge, it is not constructed. A more flexible approach would use knowledge as a guide for suggesting new features while retaining the ability to construct features unanticipated by the domain knowledge. For example, when the domain-knowledge filtering constraints were found to be too severe in the second experiment, the constraints might have been relaxed to permit the construction of additional features. This idea is evident in MIRO [Drastal and Raatz, 1989] where domain knowledge used to construct new features is retracted if the active features do not permit the construction of a consistent hypothesis.

Another issue that deserves further consideration is the generalization of new features. The method used in the tic-tac-toe experiments is very primitive: a feature is translated into features having a similar level of descriptive power (*i.e.*, at the same level of generalization). A true generalization would, for example, take a feature that detects an x in the first and second squares and generalize it to a feature that detects an occurrence of two x's side-by-side anywhere on the board. Generalized features of this sort can lead to faster learning from fewer examples, but because they also increase the complexity of the search space, they require even stronger constraints.

Acknowledgments

I would like to thank Chuck Anderson, Gunnar Blix, Carl Kadie, Gregory Piatetsky-Shapiro, Larry Rendell, Hong Shinn, Bernard Silver, and Richard Sutton for their helpful comments and suggestions during preparation of this paper.

References

Drastal, George and Raatz, Stan 1989. Learning in an abstract space. Technical Report DCS-TR-248, Department of Computer Science, Rutgers University.

Matheus, Christopher J. and Rendell, Larry A. 1989. Constructive induction on decision trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI. 645–650.

Matheus, Christopher J. 1989. *Feature Construction: An Analytic Framework and An Application to Decision Trees.* Ph.D. Dissertation, University of Illinois at Urbana-Champaign.

Muggleton, Steve 1987. DUCE, an oracle based approach to constructive induction. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence.* 287–292.

Pagallo, Giulia 1989. Learning DNF by decision trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI. Morgan Kaufmann Publishers, Inc.

Quinlan, J. Ross 1983. Learning efficient classification procedures and their application to chess end games. In *Machine Learning: An Artificial Intelligence Approach.* Tioga Press, Palo Alto. 463–482.

Quinlan, J. Ross 1987. Simplifying decision trees. *International Journal of Man-Machine Studies* 27:221–234.

Rendell, Larry A. 1985. Substantial constructive induction using layered information compression: Tractable feature formation in search. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence.* 650–658.

Schlimmer, Jeffrey C. 1987. Incremental adjustment of representations in learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, Irvine, CA. Morgan Kaufmann Publishers, Inc. 79–90.

Segre, Alberto, editor 1989. *Sixth International Workshop on Machine Learning.* Morgan-Kaufmann, Ithaca, NY.

Utgoff, Paul E. 1986. Shift of bias for inductive concept learning. In *Machine Learning: An Artificial Intelligence Approach, Vol II.* Morgan Kaufmann Publishers, Inc., Los Altos, CA. 107–148.