Utilizing Prior Concepts for Learning

Piew Datta and Dennis Kibler
Information and Computer Science Department
University of California, Irvine
Irvine, CA 92717-3425
pdatta@ics.uci.edu

Abstract

The inductive learning problem consists of learning a concept given examples and non-examples of the concept. To perform this learning task, inductive learning algorithms bias their learning method. Here we discuss biasing the learning method to use previously learned concepts from the same domain. These learned concepts highlight useful information for other concepts in the domain. We describe a transference bias and present M-FOCL, a Horn clause relational learning algorithm, that utilizes this bias to learn multiple concepts. We provide preliminary empirical evaluation to show the effects of biasing previous information on noise-free and noisy data.

1 Introduction

Traditionally computational biases have taken the form of either a representation bias or a search bias for the machine learning community. Inductive learning systems have focused on these biases for learning single concepts. Although these static biases were identified, they could not be changed since they were designed into systems. In later work, parametric algorithms allowed the user to manually change the bias before learning (Michalski, 1983). More recent work (Utgoff, 1986; Schlimmer, 1987; Rendell, Seshu, & Tcheng, 1987; Gordon, 1989; De Raedt & Bruynooghe 1992) uses biases that change the representation language or the search algorithm dynamically at execution time to accommodate the characteristics of the domain.

Although the machine learning community has focused on representation and search biases, the psychological community has identified another type of bias which we call a structural bias. The search and representation biases used in the machine learning community deal with learning a single concept. Markman's (1989) structural bias is the assumption that concepts may share a common structure. Since human categories tend to be organized taxonomically and hierarchically, this gives anyone attempting to learn the characteristics of an object a set of several possible inferences about the structure of the concept. When objects are being categorized, they inherit all or some of the characteristics of their superordinate or parent taxonomic group. Thus objects with the same parents or ancestors share some common characteristics. For example, if we know certain characteristics about house cats and we know that cats and tigers share taxonomic ancestors, then we can make some inferences about tigers, namely that they breathe, reproduce, and so on. The hierarchy of classes and the inheritance properties that develop from them is a bias that aids the learning process and allows information from one concept to be used for learning other concepts.

In this paper we develop a transference bias that carries some of the same flavor as the structural bias, but relates to inductive learning. A transference bias assumes that pieces of previously learned concepts are useful for learning other concepts in the same domain. For this bias to be useful, several concepts must exist in the domain for a transference to take place. This bias relies on the premise that concepts from the same domain share some content, and therefore, subconcepts that are frequently used for learning concepts will be used again for learning subsequent concepts.

Standard approaches to learning depend on the information available during the learning process. EBL methods are preferred if a detailed theory of the domain exists, whereas SBL methods work well if an abundance of examples exist for the domain. Hybrid methods are used when both a theory and data for the domain are available. In situations where few examples exist and the domain theory is far from complete, EBL and SBL systems will have problems learning. In these situations, a transference bias can utilize the available theory and examples. As long as some of the concepts can be learned from the available theory and examples, the remaining concepts can be learned by using the transference bias, that is, using information

learned from the previous concepts. The transference bias can be useful for learning those concepts that have few examples and are not covered by the theory.

Our goals for this paper are two-fold. First, we will introduce the idea of learning multiple concepts for inductive learning systems. Much of the work on dynamic biases has been restricted to learning a single concept. We are extending this work to learn multiple concepts in the same domain. Secondly, we utilize the transference bias for our inductive learning program. One way for illustrating this transference bias is implemented in Multiple concept FOCL (M-FOCL). M-FOCL is a relational concept learning algorithm that learns multiple concepts sequentially, biasing or utilizing knowledge that has shown to be useful in learning prior concepts. It dynamically changes its background knowledge thereby changing the representation language it uses to describe concepts.

In Section 2, we review two relational learning algorithms FOIL and FOCL, on which analysis of M-FOCL is based. Section 3 explains in greater detail the transference bias and describes M-FOCL. Section 4 evaluates the effectiveness of the transference bias by testing M-FOCL on learning legal chess moves with noise-free and noisy data. We contrast our transference bias with other types of dynamic biases in section 5. We conclude this paper by summarizing our contributions and discussing future work.

2 Relational Horn clause learning algorithms

Before describing Multiple concept FOCL (M-FOCL) and FOCL (Pazzani & Kibler, 1990), we will describe FOIL (Quinlan, 1990) to explain the core learning method applied in these systems. Given a set of extensional predicates, positive examples, and negative examples, FOIL learns a relational Horn clause concept using these predefined predicates. This concept covers all of the positive examples and excludes all negative ones. FOIL constructs the concept by performing a hill climbing search. It begins creating a clause by finding the predicate and variablization that have the highest information gain. This predicate will cover some positive examples but will most likely also cover some negative examples. FOIL specializes the clause to exclude negative examples by adding predicates to form a conjunction of predicates. FOIL continues by adding clauses until all positive examples are covered and specializes the clauses until no negative examples are covered. The final concept is a disjunction of clauses.

FOCL follows FOIL's method for constructing concepts, but FOCL can utilize intensional predicates in the form of rules. These predicates are conjunctions or disjunctions of other predicates. These predicates allow the user to define any existing domain theory for the relational learning system, however FOCL cannot revise its domain theory. FOCL's ability to use a domain theory allows it to reuse its own output, a Horn clause description, as input for learning another concept, but it does not do so.

3 Multiple concept FOCL

The premise underlying the design of M-FOCL is that different concepts in the same domain share common content, and thus information gained from learning a concept can be transferred when learning other concepts. When learning multiple concepts, M-FOCL adds the learned concepts to its domain knowledge. According to our premise these additional concepts should facilitate learning subsequent concepts. This allows M-FOCL to learn the concept and the types of information that are important in the domain. Thus by biasing the algorithm towards previously learned information, an increase in accuracy and efficiency will occur.

Currently M-FOCL takes advantage of the content of previously learned concepts in three different ways. Suppose that M-FOCL has learned the concept C defined in a pseudo-Prolog below. We use C_i to represent clauses and P_{ij} to represent predicates. To simplify the description, we have not included the variables associated with C, C_i, or P_{ij}.

$$\begin{aligned} C &:- P_{11}, P_{12}, \ldots, P_{1n_1} \equiv C_1 \\ C &:- P_{21}, P_{22}, \ldots, P_{2n_2} \equiv C_2 \\ &\vdots \\ C &:- P_{k1}, P_{k2}, \ldots, P_{kn_k} \equiv C_k \end{aligned}$$

The first method, the *entirety rule*, adds C as an intensional predicate to the domain knowledge. C is a disjunction of all clauses C_1 through C_k. The second method, the *full conjunct rule*, adds a clause, C_i, for some i, to the language as an intensional predicate. The third method, the *partial conjunct rule*, adds a conjunction of predicates as an intensional predicate to the language. The conjunction consists of at least two predicates, P_{ix} and P_{iy}, that are also used in another clause C_j where $j \neq i$. The single difference between FOCL and M-FOCL is that M-FOCL uses these predicates during induction. By adding these intensional predicates to the language, M-FOCL can perform intensional induction on examples.

The entirety rule is useful in situations where the concept to be learned is a generalization of a previous concept. For example, when learning legal chess moves, a queen move subsumes a rook move. By applying the entirety rule, M-FOCL can use a rook move to define part of a legal queen move. The full conjunct rule creates a predicate from a clause and is useful when concepts share partial information. For instance, if a queen move is already defined, when learning a rook

move, we can use one of the clauses used in the queen concept that allows the queen to move vertically on the board as part of the definition of the rook. Lastly, the partial conjunct rule is useful when concepts share some content, but do not share entire concepts or clauses. For example, since each piece must be on a board position, each clause must represent this constraint. This constraint is not represented in a concept or a clause, but is a conjunction of some predicates. By using the partial conjunct rule, a predicate is created that is the conjunction of other predicates that form useful partial concepts.

To aid the explanation of the rules, an analogy can be drawn between learning new predicates in M-FOCL and learning macros in planning systems. Macros are learned from previous successful sequences of operators (Ruby & Kibler, 1991), whereas the entirety, full, and partial conjunct rules facilitate learning new predicates from previous concepts. In the same sense that macros aid the planning system to take larger steps in the hill climbing search, creating new predicates, using these rules, helps M-FOCL take larger steps in its hill climbing search, thereby increasing the learning rate.

These rules also increase the accuracy of learned concepts by transferring information from one concept to another. The examples that were used to create concepts are indirectly used to learn other concepts. When learning with noisy data, indirectly utilizing past examples has the effect of generating more examples to learn the concept and expose noisy examples.

Using these three rules also allows M-FOCL to do less search for relevant predicates than FOCL. Each time FOCL needs to add a predicate to the clause it must search through all of the predicate variablizations. Suppose FOCL is learning a concept D that has as one of its clauses C_1 (defined earlier) from a concept learned previously. Since FOCL does not use previous concepts, it would search through all predicates and variablizations for each predicate P_{1i} in C_1, doing n_1 searches. FOCL would relearn the clause it learned earlier. M-FOCL would use the full conjunct rule to make C_1 into an intensional predicate. When it learns D, it would search the predicates once and find that C_1 has the highest information gain. It would then use C_1 instead of relearning the clause like FOCL. The effect of using previously learned subconcepts is to do selective look ahead in the hill climbing search. If the selection is good, improved performance will result. Creating intensional predicates often reduces the depth of the search when a good selection is chosen, but it also increases the branching factor of the search. Therefore, the trade-offs for generating intensional predicates with these rules must be analyzed. This is analogous to the utility problem (Minton & Carbonell, 1987) while learning macros for planning.

In accordance with the transference bias, we developed the entirety, full conjunct, and partial conjunct rules. These rules facilitate M-FOCL's use of prior concepts to learn subsequent concepts. The transference of information and the three rules increase the accuracy of the learning process. They also may decrease search since M-FOCL uses previously learned information. Lastly, the partial conjunct rule helps the algorithm find conjunctions of predicates that are used in many concepts and clauses.

4 Experiments

To experimentally evaluate the usefulness of a transference bias, we tested FOCL without any domain theory against M-FOCL. In this preliminary study we use a familiar domain, that of learning chess moves. The learning problem in this domain is similar to that discussed in Vere (1977) and De Raedt & Bruynooghe (1992) but consists of constructing concepts of legal rook, queen, knight, and bishop moves. The examples given to both of the learning systems consist of the current position and the new position the piece will be moving to. The 4-tuple $(rank1, file1, rank2, file2)$ is used to define the initial and final positions of a piece. The legal moves are defined as if no other pieces are on the board.

Two experiments were done considering various orderings of the concepts to be learned in this domain. The first measures the accuracy and performance of FOCL and M-FOCL when learning rook, knight, queen, and bishop moves. In the second experiment, we introduce classification and tuple noise into the examples to test the robustness of the mechanism. The accuracy is measured by taking the final learned concept from each algorithm, and testing the concept on a set of randomly generated examples. The percentage of these examples classified correctly is the accuracy of the algorithm. The performance of an algorithm is measured by the number of predicate variablizations tested when learning a concept which is a good measure of the amount of search these algorithms must perform during learning.

The tests for both of the experiments were conducted in the following manner. When FOCL was used, a concept representing each chess move was learned, tested for accuracy, and then discarded. The performance of FOCL was noted and then the next chess move was learned. When M-FOCL was tested, no domain theory was initially used. After it learned a move, the concept was tested for accuracy, and the performance of M-FOCL was noted. The concept was then used for subsequent learning.

FOCL was run without any domain theory as a comparison for learning each of the legal move concepts independently. For experiments with and without noise, M-FOCL was run in three different modes. In the first mode, M-FOCL only uses the entirety rule (ER) to create predicates. In the second mode, it uses the full

and partial conjunct rules (CR). The third mode uses both methods ER and CR (ER+CR).

One trial consists of learning the four concepts, rook, queen, bishop, and knight, by applying each of the methods (FOCL, ER, CR, and ER+CR) on one set of random examples. Concept ordering affects the performance of M-FOCL, since it has different domain knowledge depending on concepts learned. Experiments show benefits with M-FOCL regardless of concept ordering. However, to discount the effects of concept ordering, 50% of the possible 24 orderings were chosen randomly for each of the trials. The 12 concept orderings were selected randomly, and the M-FOCL methods were run on each of the orderings. The accuracy and performance of each of these concept orderings were then averaged together to give the results for one trial.

4.1 Learning Noise-Free Chess Moves

In this experiment, the examples given to FOCL and M-FOCL are noise-free, however, we vary the training size to see their effects on system performance and concept accuracy. According to our premise, M-FOCL should perform more accurately and efficiently then FOCL since it can use information from prior concepts. In effect M-FOCL uses the examples for a concept and the examples from prior concepts by applying the three rules. We used three sizes of training sets: 10, 25, and 50 of each positive and negative examples for each concept. We tested the training sets of size 10 and 25 with a testing size of 100 positive and 100 negative noise-free examples. The size 50 training set was tested with 200 positive and 200 negative noise-free examples. This experiment was run with 100 trials with each of the training sets on FOCL, ER, CR, and ER+CR.

Table 1: Average percent accuracy.

Training Size	Methods			
	FOCL	ER	CR	ER+CR
10	85.58	86.11	86.55	86.59
25	95.24	96.33	96.79	96.66
50	99.08	99.33	99.50	99.47

Table 1 compares the average accuracy of the four methods learning one chess move after 100 trials. With increasing training size, all four methods have better accuracy. All of the M-FOCL methods are more accurate than FOCL on all training sizes. The CR and ER+CR methods that use partial concepts perform the best in terms of accuracy since they create more flexible predicates with fewer conjunctions. Thus they can be used more often. ER performs better than FOCL but not as well as the CR and ER+CR since it can only use entire concepts, not partial concepts. When the accuracy of each of the M-FOCL methods (ER, CR, ER+CR) are compared to FOCL, they are statistically better with at least 98% confidence[1] for all of the training sizes.

Table 2 shows the average performance results for each of the four methods when learning a chess move concept. The performance was measured by noting the number of predicate variablizations that were tested to learn a concept. The performance for ER is very similar to FOCL's performance, however, ER had better accuracy. The number of predicates for both CR and ER+CR is higher than that of FOCL when the number of trials is insufficient to guarantee a high accuracy. This occurs because M-FOCL sometimes creates predicates that do not aid concept learning. When enough examples are presented (training sizes 25 and 50), methods ER and CR perform better than FOCL and learn the concept more accurately.

Table 2: Average performance.

Training Size	Methods			
	FOCL	ER	CR	ER+CR
10	116.43	115.79	134.35	145.44
25	161.86	140.66	155.89	166.58
50	166.76	139.23	144.14	155.14

4.2 Learning Chess moves with noise

In real world situations, noise-free data is rare. Thus learning systems must operate somewhat accurately. In propositional domains, classification and attribute noise are two types of noise that have been studied (Quinlan, 1986). The definitions of classification and attribute noise for relational domains are similar to those used in Brunk and Pazzani (Brunk & Pazzani, 1991).

We measure the amount of noise for a relation in the following manner. Associated with each relation is the set of tuples which satisfy the relation. We can alter the class of a tuple or an element of the tuple. The former, classification noise, refers to data where at least one datum is incorrectly assigned to a class. For our purposes only two classes of examples exist, positive and negative. When creating classification noise in our examples, for each example there is an n% chance that the example will be classified incorrectly, where n is the classification noise. (In Brunk & Pazzani (1991) and Quinlan (1986), there is an n% chance for each example of being placed in a random class.) Altering an element of the tuple is referred to as tuple noise. When

[1] We used a one sided t-test to compare FOCL to each of the other methods. The t-test used the data from each of the trials and not the averaged accuracy shown in the tables. This method was used for all of the statistical significance tests.

creating tuple noise in the data, for each element in a tuple there is an n% chance that the element's value is assigned randomly, where n is the amount of tuple noise in the data.

4.3 Classification Noise

In this noise experiment, three different values were used for the classification noise, 5%, 10%, and 20% for each trial. A training size of 50 positive and 50 negative examples, and a testing size of 200 positive and 200 negative correct examples was used. For each of the percentages of noise, we ran 100 trials on each of the four methods: independent FOCL, M-FOCL with the entirety rule (ER), M-FOCL with full and partial conjunct rules (CR), and M-FOCL with both ER and CR (ER+CR)[2].

As in the noise-free experiment, M-FOCL methods (ER, CR, and ER+CR) learned concepts more accurately than FOCL (refer to Table 3). Methods CR and ER+CR perform very closely on accuracy, although in most cases, CR was more accurate. CR and ER+CR both use partial concepts from previous concepts to aid them in learning more accurately. Sometimes these partial concepts were applied for all of the concepts learned in the domain; they can be thought of as regularities in the domain, such as not moving off of the board. This type of partial concept is what helped CR and ER+CR perform more accurately. Using a one sided t-test to compare the accuracies of the methods, we found that CR and ER+CR perform better than FOCL with 98% confidence. The t-test shows that ER's accuracy is very similar to FOCL in this experiment.

Table 3: Average percent accuracy.

Class Noise	Methods			
	FOCL	ER	CR	ER+CR
5%	90.99	91.11	91.95	91.82
10%	85.33	85.38	86.53	86.44
20%	75.73	76.02	76.93	76.94

Although these partial concepts are helpful in terms of accuracy, they can degrade performance when applied to noisy data. With the increase in classification noise, FOCL and M-FOCL tend to fit the noisy data, creating more clauses to cover the noisy positive examples. Since M-FOCL creates predicates from previous concept clauses, it generates unnecessary predicates from fitting the noise. These unnecessary predicates were then checked when learning subsequent concepts, increasing the branching factor of the hill climbing search. So predictably, the performance of ER was about 5% to 7% less efficient than FOCL. CR and ER+CR tested anywhere from 1.8 to 2.8 times more predicates than FOCL, although CR still tested fewer predicates than ER+CR.

[2]Neither FOCL nor any of the M-FOCL methods used any of the noise-tolerant abilities designed into FOCL for either of the noise experiments.

4.4 Tuple Noise

This experiment is similar to the classification noise experiment except that the amount of tuple noise is varied and no classification noise is presented. Examples were generated with 5%, 10% and 20% tuple noise and 80 trials were run on the four methods. The concept was tested for accuracy on 200 positive and 200 negative correct examples. Note that if an example attribute is noisy because of tuple noise, this does not necessarily misclassify the example.

Table 4 shows the accuracy of the four methods with varying attribute noise on 80 trials. The M-FOCL methods performed more accurately than FOCL on all three noisy sets of data. CR and ER+CR learned the concepts more accurately. The difference in accuracy between CR and FOCL increased with noisier data. As in the classification experiments, partial concepts were more useful than entire concepts when learning chess moves. The performance, in terms of the number of predicates and variablizations tested, was similar to the classification experiment. FOCL performed more efficiently than the M-FOCL methods, although ER was only slightly less efficient (.8% to 2% more predicate variablizations tested) and more accurate. CR and ER+CR were the least efficient, testing from 1.7 to 2.4 times more predicates than FOCL. Both CR and ER+CR's accuracy tested better than FOCL's accuracy with 99% confidence, whereas ER's accuracy tested at a much lower percentile[3].

Table 4: Average percent accuracy.

Tuple Noise	Methods			
	FOCL	ER	CR	ER+CR
5%	87.48	87.87	88.96	88.99
10%	81.30	81.67	83.38	83.43
20%	74.89	75.53	78.04	77.94

5 Related Systems

Our premise about the utility of a transference bias can be related to two paradigms in learning, namely dynamic biases for inductive learning and learning by analogy. This section will expand on each of these

[3]Perhaps increasing the number of trials for the classification and tuple noise experiments could increase the confidence level for ER being statistically different than FOCL. The noise experiments took about 500 hours each which restricted the number of trials that we could perform.

areas, highlighting differences between past systems and M-FOCL.

Previous work on dynamic bias include adjusting the representation language and the learning algorithm. Past systems using dynamic biases often began searching with a restricted representation and iteratively changing their representation until a hypothesis is consistent with their observations (Utgoff, 1986; Schlimmer, 1987; De Raedt & Bruynooghe, 1992). Other dynamic biasing systems modify their inductive algorithm by using a table to store the best parameter settings for various types of problems (Rendell, Seshu, & Tcheng, 1987).

M-FOCL differs from most of the systems utilizing a representation bias to aid learning. Unlike STABB (Utgoff, 1986), Clint (De Raedt & Bruynooghe, 1992), STAGGER (Schlimmer, 1987) and VBMS (Rendell, Seshu, & Tcheng, 1987), M-FOCL does not have sets of predefined languages that the learning system can use if its current language is unsuitable for learning. Instead M-FOCL transfers relevant domain information obtained from previous concepts to subsequent concepts. Unlike VBMS, STABB, and Clint, M-FOCL does not "iterate" until it finds a suitable bias, but it uses the biases made available from previous concepts and chooses the one best suited for the concept it is learning using the information gain metric. M-FOCL biases transference of content between concepts in the form of partial concept solutions rather than biasing a particular language representation or inductive algorithm as did previous systems with a dynamic bias. In addition, it only uses previous knowledge or subconcepts that have proven to be helpful in more than one situation.

FRINGE (Pagallo & Haussler, 1990) learns Boolean functions using decision trees, but iteratively modifies its tree by adding Boolean combinations of features found at the fringe of the tree as a new feature to its representation language. Since FRINGE learns only one concept, it transfers information, in the form of new features, from one part of the concept to another, avoiding the replication problem (Pagallo & Haussler, 1990). Although M-FOCL also has this self-transference ability, it focuses on transferring information from several concepts to other concepts. In addition, FRINGE creates new features without knowing their utility for learning, whereas M-FOCL only creates predicates using partial concepts if they have shown to be useful.

Since M-FOCL transfers knowledge from one concept to another it can be compared to systems using analogy to learn. Utilizing a transference bias is related to using analogy for reasoning. Cia (De Raedt & Bruynooghe, 1992), Clint's component for performing analogical reasoning, uses second order schemata to aid concept learning. It mainly relies on the syntax for the problems and solutions to find other solutions for the target domain. Since predicates and arguments can be variablized, neither the argument nor the predicate must match the original concept. SME (Falkenhainer, Forbus, & Gentner, 1986) performs matching in a similar manner except it first looks for predicates with the same name then it considers its arguments. If the arguments are the entities or functions then they are considered a match hypothesis.

Although using analogy may seem similar to the transference bias we have defined, it differs in several ways. Performing an analogy between domains implies that the information from the source domain must be generalized so that matching can occur with the target domain (Falkenhainer, Forbus, & Gentner, 1986; De Raedt & Bruynooghe, 1992). Our transference bias differs from performing analogies since the exact pieces of information are kept for learning subsequent concepts. No generalization is performed on the information. In addition, to perform an analogy between domains implies that a source domain theory must exist. Our transference bias does not require any initial domain theory, but in one sense creates a domain theory by defining concepts in the domain. Lastly, analogy uses one example, that of the source domain, whereas the transference bias emphasizes learning from more than one concept in the domain.

6 Discussion

We have presented a new bias, the transference bias, that takes advantage of previously learned concepts when learning a new concept. M-FOCL is an implemented program that uses the transference bias. In particular, by using three rules for transferring knowledge from previously learned concepts, preliminary results show that M-FOCL can learn concepts more accurately, and under some circumstances, more efficiently than without this bias. Using M-FOCL to learn in the chess moves domain provides an illustration of the learning method and evidence for our transference bias premise.

The transference bias is useful in situations where a domain is only partially understood either with a weak domain theory or sparse data sets. A domain may have a strong domain theory for some concepts or it may have large amounts of examples for other concepts, but often it does not have both of these types of information for all concepts. M-FOCL compensates for the lack of a strong domain theory or large amounts of examples by using the transference bias. This bias requires that the domain has some concepts that can be learned with the initial domain theory and examples. Once this condition is satisfied, the transference bias overcomes a lack of data by presuming a regularity in the domain. Thus the transference bias fills a gap between EBL and SBL methods in that it neither needs a strong domain theory nor a large example set to learn

concepts in a domain. Domains with these characteristics are common in real world domains, making the transference bias an important topic for learning research.

We plan to continue by experimenting in more complex domains to demonstrate the generality and effectiveness of the transference bias. More specifically, we are considering testing the utility of the transference bias on natural domains, such as medical disease classification. Natural real world domains provide a large array of concepts to test our transference bias. In addition, they often have sparse data sets and weak domain theories. The transference bias overcomes these deficiencies in the data by using previously learned information for learning subsequent concepts. We also plan to develop a more theoretical understanding of the nature of transference, characterizing situations where this bias would benefit the learning process.

Acknowledgments

We would like to thank Cliff Brunk, Kamal Ali, and Mike Pazzani for their help with FOCL and commenting on an earlier draft of this paper.

References

Brunk, C. A., Pazzani, M. (1991). An investigation of noise-tolerant relational concept learning algorithms. *Proceedings of the Eighth International Workshop on Machine Learning*, Los Altos, CA: Morgan Kaufmann.

De Raedt, L., Bruynooghe, M. (1992). Interactive concept-learning and constructive induction by analogy. *Machine Learning*, volume 8, no. 2. Boston, MA: Kluwer Academic Publishers.

Falkenhainer, B., Forbus, K., Gentner D. (1986). *The structure-mapping engine.* (Technical Report no. UIUCDCS-R-86-1275). Urbana, Illinois: University of Illinois at Urbana-Champaign, Department of Computer Science.

Gordon, D. (1989). Screening hypotheses with explicit bias. *Proceedings of the Sixth International Workshop on Machine Learning.* Los Altos, CA: Morgan Kaufmann.

Markman, E. (1989). *Categorization and naming in children.* Cambridge, Massachusetts: The MIT Press.

Michalski, R. S. (1983). A theory and methodology of inductive learning. *Machine Learning: An Artificial Intelligence Approach*, volume 2. Los Altos, CA: Morgan Kaufmann.

Minton, S., Carbonell, J. G. (1987). Strategies for learning search control rules: an explanation-based approach. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence.* Milan, Italy.

Mitchell, T. M. (1980). *The need for biases in learning generalizations.* (Technical Report CBM-TR-117). New Brunswick, NJ: Rutgers University, Department of Computer Science.

Pagallo, G., Haussler, D. (1990). Boolean feature discovery in empirical learning. *Machine Learning*, volume 5, no. 1. Boston, MA: Kluwer Academic Publishers.

Pazzani, M. J., Kibler, D. (1990). *The utility of knowledge in inductive learning.* (Technical Report no. 90-18). Irvine, CA: University of California, Irvine, Department of Information and Computer Science.

Quinlan, J. R. (1986). The effect of noise on concept learning. *Machine Learning: An Artificial Intelligence Approach*, volume 2. Los Altos, CA: Morgan Kaufmann.

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, volume 5 no. 3. Boston, MA: Kluwer Academic Publishers.

Rendell, L., Seshu, R., Tcheng, D. (1987). More robust concept learning using dynamically-variable bias. *Proceedings of the Fourth International Workshop on Machine Learning.* Los Altos, CA: Morgan Kaufmann.

Ruby, D., Kibler, D. (1991). SteppingStone: An empirical and analytical evaluation. *Proceedings of the Ninth National Conference on Artificial Intelligence.* Los Altos, CA: Morgan Kaufmann.

Schlimmer, J. C. (1987). Incremental adjustment of representations. *Proceedings of the Fourth International Workshop on Machine Learning.* Los Altos, CA: Morgan Kaufmann.

Utgoff, P. (1986). Shift of bias for inductive concept learning. *Machine Learning: An Artificial Intelligence Approach*, volume 2. Los Altos, CA: Morgan Kaufmann.

Vere, S. A. (1977). Induction of relational productions in the presence of background information. *Fifth International Joint Conference on Artificial Intelligence.* Los Altos, CA: Morgan Kaufmann.