Machine Learning, 15, 5–24 (1994)

Incremental Abductive EBL

WILLIAM W. COHEN (WCOHEN@RESEARCH.ATT.COM)
AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

Editor: Paul Utgoff

Abstract. In previous work, we described a knowledge-intensive inductive learning algorithm called abductive explanation-based learning (A-EBL) that uses background knowledge to improve the performance of a concept learner. A disadvantage of A-EBL is that it is not incremental. This article describes an alternative learning algorithm called IA-EBL that learns incrementally; IA-EBL replaces the set-cover-based learning algorithm of A-EBL with an extension of a perceptron learning algorithm. IA-EBL is in most other respects comparable to A-EBL, except that the output of the learning system can no longer be easily expressed as a logical theory. In this article, IA-EBL is described, analyzed according to Littlestone's model of mistake-bounded learnability, and finally compared experimentally to A-EBL. IA-EBL is shown to provide order-of-magnitude speedups over A-EBL in two domains when used in an incremental setting.

Keywords. inductive learning, combining empirical and analytical learning, pac-learning, explanation-based learning, abduction

1. Introduction

In previous work, we described a hybrid explanation-based/inductive learning algorithm called abductive explanation-based learner (A-EBL) that uses background knowledge to improve the performance of a concept learning (Cohen, 1992a). This learning system was shown to satisfy Valiant's criterion of pac-learnability (Valiant, 1984) and was also empirically validated on two learning problems that arose in the context of learning from the information and examples in a textbook on the game of contract bridge.

A disadvantage of A-EBL is that it is not incremental. If new training examples become available after a learning episode, there is no way to adjust the hypothesis of the learner incrementally to account for the new data; the only way of using the new data is to add them to the old set of training examples and re-run the learning system. This may be a disadvantage in some settings. In particular, incremental learning is useful for "serial learning tasks" (Utgoff, 1989), in which a stream of training instances is provided, rather than a single fixed set of instances; in this case, repeatedly revising an existing hypothesis may be less expensive than repeatedly re-forming a hypothesis from scratch when each new instance appears. Another motivation for considering incremental learning algorithms is that they are, in general, more plausible as models of human learning (Langley et al., 1987).

This article describes an alternative learning algorithm called IA-EBL that learns incrementally. IA-EBL replaces the set-cover based learning algorithm of A-EBL with an extension of a perceptron learning algorithm. In most respects, IA-EBL is equivalent to A-EBL; in particular, it is competent for the same class of problems as A-EBL and has similar convergence properties, and thus inherits all of the strengths (and weaknesses) of

A-EBL. However, IA-EBL can update its hypothesis efficiently in response to new data, and is therefore more efficient in incremental settings.

In this article, IA-EBL is described, analyzed according to Littlestone's model of mistake-bounded learnability, and compared empirically to A-EBL. The analysis and the experimental results show that the hypotheses of IA-EBL are comparable to those of A-EBL in predictive accuracy. The main disadvantage of IA-EBL is that, unlike A-EBL, the hypotheses of IA-EBL cannot be expressed easily as logical theories; thus hypotheses are in general less perspicuous.

2. Background

2.1. *Description of A-EBL*

A-EBL is a learning algorithm that solves the *theory specialization problem* (Flann & Dietterich, 1989):

Given: 1) an overgeneral theory T_0,
2) examples of correct/incorrect predictions made using T_0,

Find: a theory T_s that specializes T_0, and only leads to correct predictions.

The specialization T_s of the initial theory T_0 will be called the *target theory.* A-EBL forms a target theory by using EBG on the initial theory T_0 and the examples to generate a small set of rules that explains all the positive data. This small set of rules becomes the new specialized theory T_s, and the original theory is discarded. As an example of an application of this technique, A-EBL has been used to specialize an initial theory T_0 for the concept "plausible opening bid" into a theory T_s for the concept "correct opening bid" using the examples given in a textbook on contract bridge.

A brief outline of the A-EBL algorithm is shown in table 1. A-EBL takes as input a set of positive examples S^+, a set of negative examples S^-, a theory T_0, and an operationality predicate $\mathcal{O}$. First, all possible generalizations of the positive training examples are enumerated, where "all possible generalizations" means all generalizations formed by applying EBG to some explanation structure of a positive example. (We use the phrase "*some*

Table 1. The A-EBL learning algorithm.

Algorithm A-EBL(S^+, S^-, T_0, $\mathcal{O}$):

1. Compute the explanation structure of every proof in T_0 of every example in S^+.
2. For each explanation structure found in step 1, find the set of candidate rules that can be formed by applying the final stage of EBG to the resulting explanation structure, i.e., by replacing constants with variables in the explanation structure and then extracting a rule from the generalized structure.
3. Filter the set of candidate rules by removing any rule that covers an element of S^-.
4. Use a greedy set cover to find a small set of the remaining candidate rules that covers all of the examples in S^+.
5. Return that set of rules as the theory T_s.

explanation structure" rather than "*the* explanation structure" because A-EBL allows one to have multiple explanations for each example. A-EBL is thus suited to initial theories that suffer from the *multiple explanation problem* (Rajamoney & DeJong, 1988; Pazzani, 1988).) This phase of the algorithm requires the initial theory T_0 to be *tractable* in the sense that all explanations of every example can be generated easily. Inconsistent generalizations (those that match some negative example) are then filtered out, and finally, a greedy set cover algorithm is used to find a small set of the remaining candidate rules that covers all the positive examples.

Notice that the rules found by A-EBL are not added to the original theory T_0, but are pulled out into a separate theory T_s; the intention is that T_s will model the data more accurately than the original overgeneral theory T_0. One can think of A-EBL as learning by "throwing away" the incorrect parts of an overgeneral theory; in this respect, A-EBL is similar to Flann and Dietterich's (1989) IOE technique and is an example of the use of EBL for *knowledge-level learning*, as discussed by Rosenbloom and Aasman (1990).

2.2. Discussion of A-EBL

We should note that A-EBL has several limitations that will not be addressed in this article. One limitation is that it must be possible to generate *all* explanations of each example. There are interesting classes of first-order theories that do have this property—the class of circuit diagnosis theories described in section 5.2.1 is one example—but unfortunately, it is hard to restrict initial theories syntactically so that this property must hold without eliminating large classes of potentially interesting theories. For example, even propositional Horn-clause theories can have an exponential number of proofs. The practical implication of this fact is that the initial theory must, to some degree, be engineered by the user.

Another limitation is that the problem A-EBL solves, theory specialization, is a special case of the more general problem of *theory revision* (Ourston & Mooney, 1990; Pazzani et al., 1991; Towell et al., 1990), in which the initial theory is assumed to be overly general. It might be asked why we choose to study this special case, rather than the general one. However, as we have argued previously (Cohen, 1991), the type of theory specialization performed by A-EBL is more general than it appears; in particular, A-EBL's style of theory specialization is also useful for several other types of knowledge-based learning tasks, including "Induction over the Unexplained" (Mooney & Ourston, 1989), theory completion (Danyluk, 1989; Fawcett, 1989; Ali, 1989), theory-based constructive induction (Drastal et al., 1989), and learning from a weak theory containing determinations (Mahadevan, 1989). The ability to solve many types of restricted knowledge-based learning problems is a useful complement to the ability to solve very general knowledge-based learning problems such as general theory revision. This is particularly true given recent evidence that special-purpose theory revision systems can dramatically outperform general-purpose ones on some learning tasks (Cohen, 1992b).

A second reason for considering theory specialization is that it is more tractable analytically than the general case of theory revision. We are thus able to develop a rigorous understanding of theory specialization algorithms, which may ultimately shed light on the general problem.

3. A formal model of incremental learning

Our major formal result is to present a provably efficient incremental version of A-EBL; the efficiency of this algorithm is proved relative to the model of efficient incremental learnability introduced by Littlestone (1988). In this model, at each time step i, the following sequence of events occurs:

1. The learning algorithm *ILEARN* receives a new example x_i.
2. *ILEARN* then makes a *prediction* of 0 or 1. This prediction is a guess of the value of $\mathfrak{C}_{T_s}(x_i)$, the correct classification of x_i with respect to the target concept T_s.[1]
3. Finally, *ILEARN* receives the correct classification of x_i.[2] This classification is used to *update* the rules used in step 2 to make a prediction.

The learnability model is summarized in figure 1. Notice that previous examples are not stored (except implicitly, in the rules used in step 2 to make a prediction). As the figure shows, one natural implementation of an incremental learning algorithm is as a pair of functions $\langle$*predict*, *update*$\rangle$ together with a standard protocol for using these routines. The protocol must require that *predict* is used only to predict the classification of a new example x_i, and *update* is used only to adjust incrementally the rules used by *predict*.

How can one evaluate such a learning system analytically? It is relatively easy to analyze the time that an incremental learner will require to process each x_i. The harder task is to determine the number of examples required to learn a target concept. One problem is that if an adversarial sequence of examples is presented (for example, if some x_i is repeated an arbitrary number of times), exact identification of the target may take place only at the limit. One way to avoid this problem is to evaluate the learner by measuring the worst-case number of prediction errors or *mistakes* the learning algorithm makes en route to exact identification. This measure is called the *mistake bound* of the learner.

More precisely, define a *presentation of* T_s to be a (possibly infinite) sequence of pairs $p = \langle x_1, \mathfrak{C}_T(x_1)\rangle, \ldots, \langle x_i, \mathfrak{C}_T(x_i)\rangle, \ldots$, and let *MISTAKES*(*ILEARN*, p) denote the

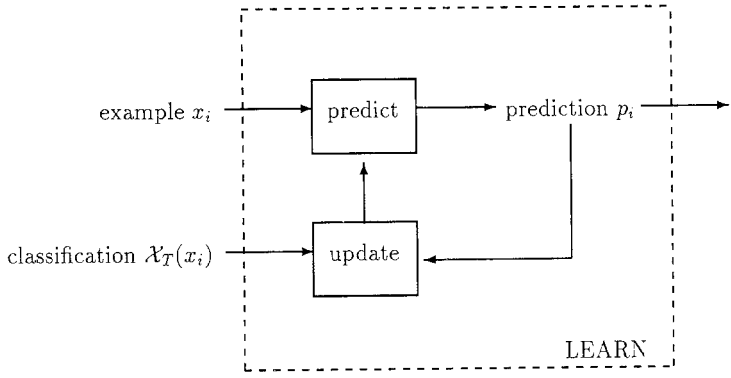

Figure 1. An incremental learning system.

number of prediction mistakes made by *ILEARN* given the presentation p. Finally, let $\mathcal{HC}$ be some space of possible target concepts, and let $\mathcal{HC}_n$ denote the set $\{h \in \mathcal{HC} : size(h) \leq n\}$. The function $b(n)$ is a *mistake bound for ILEARN and the concept class* $\mathcal{HC}_n$ if and only if for all $T_s \in \mathcal{HC}_n$, and for all presentations p of T_s,

$$MISTAKES(ILEARN, p) \leq b(n)$$

In the remainder of this article, the mistake bound of an incremental learner will be used to evaluate its efficiency: an efficient incremental learner will be one that spends polynomial time on each input x_i, and has a polynomial mistake bound.

It has been shown (Littlestone, 1988, 1989a) that the mistake bound of an incremental learner is closely related to the number of examples required to achieve predictions that are *probably approximately correct* (pac) (Valiant, 1984). In particular, existence of a polynomial mistake bound for a concept class $\mathcal{HC}$ implies existence of a pac-learning algorithm with a polynomial sample complexity. The converse, however, is not true (Blum, 1990); thus, mistake-bounded learnability is a strictly stronger model than pac-learnability.

4. An incremental version of A-EBL

One class of incremental learning methods that can learn disjunctions are the various perceptron learning methods. Littlestone has described several perceptron learning algorithms with polynomial mistake bounds (Littlestone, 1988, 1989b). Since A-EBL learns a theory that is essentially a disjunction of EBG rules, it seems plausible that these perceptron algorithms can be adapted to the A-EBL learning problem. The main obstacle is that perceptron learning algorithms require all the features of the disjunction to be known in advance. For A-EBL, the "features" that are to be disjoined are rules created by using EBG on a fixed initial theory, and it may be impossible—and is generally speaking impractical—to enumerate all of these in advance.

We have extended the WINNOW1 perceptron learning algorithm (Littlestone, 1988) to handle the A-EBL learning problem. The WINNOW1 algorithm was chosen because it is easy to analyze and is relatively insensitive to the number of potential features; the latter property is crucial in our setting, where the number of potential features is large or (in the case of a recursive theory) infinite.

Table 2 shows our extension, which we call IA-EBL. The IA-EBL algorithm associates a weight with each possible rule R; by default, this weight is proportional to $2^{-size(R)}$, where $size(R)$ denotes the number of nodes in the explanation structure used to create the rule R.[3] For each input x_i, the algorithm computes the set $\mathcal{E}_{x_i}$ of all candidate rules that explain x_i, and then predicts that x_i is in the target concept if and only if the sum of the weights of the candidate rules is greater than the threshold value $\frac{1}{2}$. If the prediction is wrong, the weights of the candidate rules are either increased (by doubling) or set to zero, depending on whether the sum was too large or too small. Setting a weight to zero means that the rule will have no later effect, and is effectively the same as discarding the rule. In the limit, only consistent explanations will have a nonzero weight, and those that support many positive examples will have the largest weights. An example will be predicted to be in the target concept if there are enough strongly weighted candidate rules.

Table 2. The IA-EBL learning algorithm.

In the code below,

- $\mathcal{E}_x$ is the set of candidate rules for an example x as computed in steps 1 and 2 of A-EBL.
- W is a set of ordered pairs, initially empty, shared by *predict* and *update*. Conceptually, the pairs in W are used to define a mapping from a rule R to a real-number value *weight*(R), which is computed as follows:

$$weight(R) \equiv \begin{cases} w & \text{if } \langle R, w \rangle \in W \\ 2^{-size(R)\cdot\log_2(|T_0|+1)} & \text{otherwise} \end{cases}$$

where $|T_0|$ denotes the number of rules in the initial theory T_0.

Algorithm IA-EBL(x_i, $\mathfrak{T}_{T_s}(x_i)$, T_0, Θ):

Predict: Return the prediction

$$p_i = \begin{cases} 1 & \text{if } \sum_{R \in \mathcal{E}_{x_i}} weight(R) \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

Update:

- If the prediction was 0 and the correct classification is 1, then for each $R \in \mathcal{E}_{x_i}$, double *weight*(R) by replacing W with

$$W - \{\langle R, weight(R)\rangle\} \cup \{\langle R, 2 * weight(R)\rangle\}$$

- If the prediction was 1 and the correct classification is 0, then for each $R \in \mathcal{E}_{x_i}$, zero *weight*(R) by replacing W with

$$W - \{\langle R, weight(R)\rangle\} \cup \{\langle R, 0\rangle\}$$

The procedure for assigning weights may seem somewhat unmotivated; however, the basic idea is not counterintuitive. First, rules are initially given very small weights to ensure that the total weight of any example is within a bounded range, even if that example happens to have many proofs; this makes it easier to design update procedures that always terminate. The update procedures move weights in the "right direction" to classify an example correctly. Weights for rules that justify a negative example are reduced to zero, since they cannot be part of the target theory. It can be shown easily that this is the optimal choice, in the sense that it moves the weights as far as possible in the right direction. On the other hand, while it is clear that weights for rules justifying a misclassified positive example must be increased in some way, we cannot know how far to increase each weight, since we do not know which rules are part of the target concept (and thus should have a high weight) and which are not (and hence should have a low weight). The solution is to increase the weight of each rule by the same small amount; thus, a rule will have a large weight only if it supports many positive examples. Finally, since the weights start out exponentially small, increasing the weights by adding some constant does not work;

instead, we must multiply by a constant. The constants one half and two are somewhat arbitrary; a different version of the convergence theorem (below) could probably be proved for other constant values.

We note also that this algorithm employs a nove way of *using* EBG rules: a weight is given to each rule, and a new instance x is assumed to be a member of the learned concept only if the sum of the weights of applicable rules is over a certain threshold. This scheme is somewhat less rigid than most previous schemes for using EBG rules, including the scheme employed by A-EBL itself.

5. Results

5.1. Formal results

The main formal result of this article is the following theorem, which shows that IA-EBL has a polynomial mistake bound on the class of problems solvable by A-EBL.

Theorem 1. *Let $\mathfrak{IC}$ be the set of theories T_s that each contain a set of rules $\{R_1, \ldots, R_k\}$ such that every R_i is in the candidate set of rules generated by A-EBL in steps 1 and 2 for some sample S^+; that is, let $\mathfrak{IC}$ be the set of target theories learnable by A-EBL. Let the size measure on some $h \in \mathfrak{IC}$ be defined as*

$$size(\{R_1, \ldots, R_k\}) = \sum_{i=1}^{k} size(R_i)$$

where $size(R_i)$ is defined as the number of nodes in the explanation structure from which R_i was derived; again, this is the size measure used by A-EBL. Recall that $\mathfrak{IC}_n$ denotes the set $\{h \in \mathfrak{IC} : size(h) \leq n\}$. Finally, assume that T_0 is tractable *in the sense that all explanations for any example x_i can be generated in time polynomial in the size of x_i.*

Then IA-EBL runs in time polynomial in the size of x_i, and has a mistake bound on the concept class $\mathfrak{IC}_n$ of

$$3n \log_2(|T_0| + 1) + 2$$

Proof. In the proof, we will adopt Littlestone's terminology: a *promotion* refers to the actions taken after a false negative, and an *elimination* refers to the actions taken after a false positive. We will let E denote the number of eliminations, P denote the number of promotions, and *scale* denote the quantity $log_2(|T_0| + 1)$. Finally, we will let $\mathfrak{R}$ denote the set of all rules that could be generated as candidate rules by A-EBL for some possible sample S^+, $T = \{R_{i_1}, \ldots, R_{i_k}\}$ be the target concept, and $r = scale * size(T)$. Throughout the proof, *size* will be the size function defined above.

Clearly, the algorithm can be implemented in polynomial time. In fact, if a hash table is used for W (as in our implementation), it is linear in $|\mathcal{E}_x|$. Notice that since it is assumed that $\mathcal{E}_x$ is generated in polynomial time, it must be of polynomial size.

To obtain the desired mistake bound, the following lemma will be needed.

Lemma 1

$$\sum_{R \in \mathcal{R}} \frac{1}{2^{scale*size(r)}} \leq 1$$

Proof. In theorem 3 of Cohen (1992a), we showed that each rule R can be encoded by the explanation structure from which it was derived, which in turn is uniquely defined by the sequence of clause labelings generated by the following procedure: traverse the nodes of the explanation structure in postfix order, reading off the names of the clauses that were used to prove each term. The symbol **nil** is used to indicate when an operational leaf is reached. The result of this is a string of length n whose letters are either the names of one of the $|T_0|$ clauses in the theory, or **nil**. Using this encoding, a rule R can be encoded in $\log_2(|T_0| + 1) * size(R) = scale * size(R)$ bits.

Notice that since we can deduce the number of children of each node from the clause that labels that node, we can reconstruct the tree itself from the postfix order. Importantly, we can tell when the last node of the tree has been reached; the encoding is thus self-delimiting, and hence the set of all valid codes is prefix free.[4] The lemma thus holds by Kraft's inequality (Abramson, 1963, pp. 53). ■

The remainder of the analysis closely follows the analysis in Littlestone (1988) of the perceptron learning algorithm WINNOW1. The mistake bound itself is obtained by an amortization argument. Consider the quantity

$$Q \equiv \prod_{j=1}^{k} weight(R_{i_j})$$

where the R_{i_j}'s are the rules in the target concept $T = \{R_{i_1}, \ldots, R_{i_k}\}$. The weights $weight(R_{i_j})$ are always less than one, since 1) weights are initially less than one, 2) the only way weights increase is by promotion, and 3) prior to a promotion, the sum of weights must be less than $\frac{1}{2}$. Hence Q is always less than 1. But each promotion at least doubles Q; since Q is initially equal to

$$\prod_{j=1}^{k} \frac{1}{2^{scale*size(R_{i_j})}} = \frac{1}{2^{scale*size(R_{i_1})+\ldots+scale*size(R_{i_k})}} = \frac{1}{2^r}$$

it follows that the number of promotions is at most r, that is,

$$P \leq r \tag{1}$$

Now consider the quantity

$$S \equiv \sum_{R \in \mathcal{R}} weight(R)$$

Initially, $S \leq 1$ by the lemma. Each promotion increases S by at most 1, since promotion doubles a set of weights for which the weighted sum is at most $\frac{1}{2}$. Analogously, elimination decreases S by at least $\frac{1}{2}$, since it eliminates a weighted sum larger than $\frac{1}{2}$. Finally, S is always greater than 0; hence at all times,

$$0 < S \leq 1 + P - E/2$$

Solving for E, we get $E \leq 2P + 2$. Since $P \leq r$ by equation (1), we obtain the claimed result

$$\text{mistakes} = E + P \leq 2r + 2 + r = 3r + 2 = 3r + 2 = 3\log_2(|T_0| + 1) * size(T) + 2$$

completing the upper bound on the number of mistakes made by the algorithm. ■

A corollary of this theorem is that IA-EBL learns (with a polynomial mistake bound) any target theory that is pac-learned by A-EBL; thus, the theorem also shows that IA-EBL is at least as general as A-EBL. Using Littlestone's (1989a) results relating mistake-bounded learnability and prediction accuracy, one can also show as a corollary that if the x_i's are always chosen stochastically from a fixed distribution, then with probability at least $1 - \delta$, the predictions made by IA-EBL will have error no more than ϵ after

$$O\left(\frac{n \log |T_0|}{\epsilon} \log \frac{1}{\delta} + \frac{n \log |T_0|}{\epsilon} \log(n \log |T_0|)\right)$$

training examples have been processed. This is comparable to the bounds on sample complexity for A-EBL presented by Cohen (1992a).

To summarize, we have shown that IA-EBL, like A-EBL, has a polynomial sample complexity in the worst case, and also that it processes each example in time polynomial in the size of that example. This last point indicates that IA-EBL in the limit will be faster than A-EBL when used incrementally. Consider the following argument. Let $p_b(t)$ be the polynomial run-time of A-EBL (for a set of examples of total size t), and let $p_i(n)$ be the polynomial run-time of IA-EBL (for a single example of size n). Now imagine that we have an ordered set of examples $x_1, \ldots, x_m$, each of size n, and that we would like to process these incrementally—i.e., we would like to produce a hypothesis T_{s_i} after each prefix $x_1, \ldots, x_i$. To do this with A-EBL would require running A-EBL on the first example alone, then the first two examples, the first three, and so on: the time required for this would be

$$\sum_{i=1}^{m} p_b(i * n)$$

However, to do this with IA-EBL will require only time proportional to

$$\sum_{i=1}^{m} p_i(n)$$

Assuming that p_b is at least linear, then when $m \gg n$, the first sum grows at least quadratically. However, when $m \gg n$, IA-EBL will take only time linear in m.

5.2. *Experimental results*

Unfortunately, the analysis above does not show that IA-EBL will be faster on reasonable-sized samples, nor does it give more than rough upper bounds on what the learning rate will be. It is also possible that IA-EBL may converge much more slowly than A-EBL in the average case, even though the worst-case bounds of each are comparable. In this section, we will compare IA-EBL empirically to A-EBL to verify that it is an efficient incrementalization of A-EBL.

5.2.1. *Description of the learning tasks*

As an additional test of the efficiency and generality of IA-EBL, IA-EBL and A-EBL were compared empirically on two problems from two different domains. For both problems, the initial theory is a recursive first-order Horn clause theory. The first problem is that of learning bridge opening bids from random data, as described in Cohen (1992a). The second problem is inspired by one that has been well studied in the literature on abduction and model-based diagnosis: diagnosing a faulty circuit (de Kleer & Williams, 1987; Reiter, 1987). This problem is often solved using abductive reasoning. Here we will consider a variant of this problem to which A-EBL is well suited: using examples to predict the *behavior* of a circuit that contains an unknown fault. In addition to providing an additional benchmark for testing IA-EBL, this problem illustrates the relationship of A-EBL to traditional abductive reasoning.

In modeling the circuit domain, we will make a number of standard assumptions. First, we assume that each circuit component computes a boolean function. Second, we assume that each component is either functioning normally, or is in one of two error states: it is either stuck at 1 (i.e., its output is always 1, regardless of the inputs) or stuck at 0. Given these assumptions, a *diagnosis* for a circuit is simply an indication of the state of each component: for example, one possible diagnosis for the circuit of figure 2 would be $\{stuckat1(x1), ok(x2), ok(a1), ok(a2), ok(o1)\}$. Finally, we assume that the circuit contains at most a single fault.

Now we must construct a theory for this circuit that allows the behavior of the circuit to be predicted given a diagnosis for the circuit. First, we will specify the connectivity of the components as follows:

```
full_adder(A,B,Cin,Cout,Sum) ←
    xor_gate(Fault,x1,A,B,Xlout) ∧
    and_gate(Fault,a2,Xlout,Cin,A2out) ∧
    xor_gate(Fault,x2,Xlout,Cin,Sum) ∧
    and_gate(Fault,a1,A,B,Alout) ∧
    or_gate(Fault,o1,Alout,A2out,Cout).
```

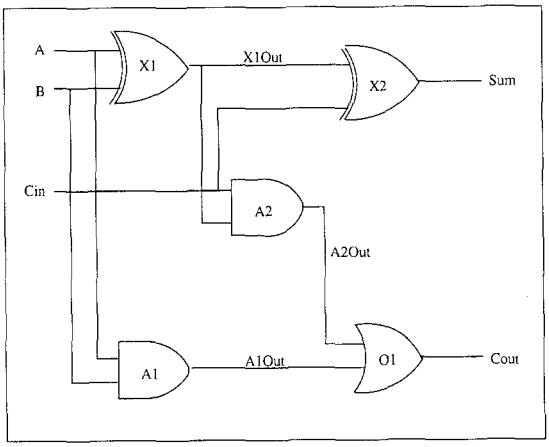

Figure 2. A circuit for a full-adder.

This specification requires that we define a series of predicates of describing the behavior of each gate type. For gates computing the function f, these predicates will have the following form:

$$f\_\text{gate}(\text{Fault},\text{GateId},\text{Input}_1,\ldots,\text{Input}_k,\text{Output}_1,\ldots,\text{Output}_n)$$

We interpret this predicate as saying: "if the gate with name *GateId* is in a circuit with the single fault *Fault*, then it will produce outputs $Output_1, \ldots, Output_n$ given inputs $Input_1, \ldots, Inputs_k$." We will give the definition of *and_gate* only; the other predicates are defined analogously.

The definition of *and_gate* requires three rules. If the gate is stuck at 1, then its output is always 1, and the fault must be *stuckat1(GateId)*; the case for stuck at 0 is analogous.

and_gate(stuckat1(GateId),Gate,In1,In2,1) ← stuckat1(GateId).
and_gate(stuckat0(GateId),Gate,In1,In2,0) ← stuckat0(GateId).

Otherwise, if the gate is normal, then it computes the boolean AND of its inputs, regardless of what the fault in the circuit is.

and_gate(Fault,GateId,In1,In2,Out) ← ok(GateId) ∧ and(In1,In2,Out).

This theory allows one to predict the behavior of a circuit given a correct diagnosis: the problem, of course, is that the diagnosis is not known. To complete the process of modeling this problem as a theory specialization task, we will add to this theory the facts *ok(G)*, *stuckat1(G)*, and *stuckat0(G)* for every gate *G*. The theory now becomes overgeneral, and can be specialized by A-EBL. Notice that in the overgeneral theory, no example can have more than $2n + 1$ explanations, where n is the number of gates in the circuit: this follows because once a value has been chosen for the single common *Fault*, the rest of

the explanation becomes fixed, and there are at most $2n$ possible faults (plus one diagnosis corresponding to a normal circuit). Thus, these circuit-diagnosis theories are also tractable.

5.2.2. *Discussion of the circuit domain*

This problem gives a good illustration of the similarities and differences between A-EBL and abduction. Abductive approaches usually produce as output the set of all simplest diagnoses consistent with the examples: the hope is that this set will contain the correct diagnosis and not too many incorrect ones. Unfortunately, abductive approaches based on set-covering are usually intractable, if for no other reason than that the set of minimal diagnoses may be exponentially large. (However, finding a single minimal set cover is also computationally hard.)

In contrast, A-EBL and IA-EBL use a polynomial-time algorithm to generate a single hypothesis. This hypothesis may or may not correspond to the correct diagnosis; indeed, it may not correspond to a valid diagnosis at all.[5] However, it can still be guaranteed that this hypothesis will predict the behavior of the system being diagnosed accurately on novel cases drawn from the same distribution as the training cases; in other words, if the diagnosis is not correct, the behavior of the system will be very nearly the same as if the diagnosis were correct.

This surprising result is echoed by results in computational learning theory in which learning is made tractable by increasing the class of possible hypotheses. For example, k-term DNF is not pac-learnable if the hypotheses of the learner are constrained to be k-term DNF formulae (Kearns et al., 1987); however, k-term DNF *is* learnable if the learner is allowed to use the larger class of k-CNF formulae as hypotheses (Valiant, 1984). In short, the difficulty in learning k-term DNF is that the search problem is overconstrained; using a larger hypothesis space makes the search tractable. Similarly, while the problem of finding a consistent diagnosis is intractable, by allowing A-EBL to generate inconsistent diagnoses as hypotheses, the problem of searching for a hypothesis that explains the data is made tractable. As in the case of k-term DNF, the hypothesis space used by A-EBL is still small enough that a consistent hypothesis will be accurate on novel examples.

5.2.3. *Methodology*

The small circuit of figure 2 is extremely easy to diagnose; we judged it too easy to form a good basis for comparing the two learning algorithms. In the experiments, we used instead an eight-bit ripple-carry adder. The examples of the behavior of this circuit were generated by simulating a circuit to which a random fault had been introduced, and were examples of the predicate *nth_output(N,ABits,BBits,Out)*, which is true if *Out* is the N-th output of the adder when given as inputs *ABits* and *BBits*; each of the inputs is a list of binary inputs. This predicate can be implemented using a theory of the sort used in the example above; notice that the theory is recursive and makes some use of function symbols. However, this problem is a hard one for diagnosis systems: since the circuit is very deep,[6] there are many different possible components whose failure might affect a given output.

For both the bridge domain and the circuit diagnosis domain, the following experiment was conducted. First, a testing set of 1000 examples was generated and set aside. Next, a training set (containing 300 examples for the bridge domain and 100 examples for the circuit domain) was generated. This training set was presented to A-EBL and IA-EBL in small "batches" (20 examples for the bridge domain and 10 examples for the circuit domain). The ordering of batches and of examples within each batch was random. After each batch was presented, A-EBL was run using the examples in that batch together with all previous batches. IA-EBL, however, was simply trained on the new batch of data until it correctly classified these new examples. The hypothesis of each system was then tested to determine its accuracy. Each experiment was repeated 10 times, and the results were averaged to obtain the learning curves in figure 3. In the circuit domain, a different random fault was introduced in each trial.

Notice that since IA-EBL adjusts its weights gradually, it may be that an example needs to be presented several times before being correctly classified;[7] this means that many passes needed to be made over each new batch of examples. In the experiments, we adopted the following heuristics for cycling through the elements of a batch. On the first cycle, each example in the batch is presented to IA-EBL in turn, and weights are updated repeatedly until that example is correctly classified. In subsequent cycles, only examples that were misclassified in the previous cycle are presented. If no examples were misclassified in a cycle, a final pass is made through the entire batch to see if any examples are misclassified, and if so, they are collected together, and the procedure described above is repeated. No batches in these experiments required more than 11 such cycles, and most required only one or two.

This ordering seemed to be faster than the obvious procedure of simply cycling through all elements repeatedly.[8] Two other tricks were used to speed up learning. First, the set of explanations of each example was cached, so that this set did not need to be recomputed in each cycle. Second, when a positive example x_i had total weight t, where $t < \frac{1}{2}$, weights were multiplied by $2^{\lfloor \log_2 \frac{1}{t} \rfloor - 1}$ rather than simply being doubled. This is precisely equivalent to presenting the example $\lfloor \log_2 \frac{1}{t} \rfloor - 1$ times, and hence does not affect the convergence result; however, it means that the classification of x_i will be corrected immediately.

In the figures, the system described above is labeled "IA-EBL (no memory)." As a variant of this procedure, we also considered a hybrid batch/incremental version of IA-EBL, labeled "IA-EBL (full memory)." This system is identical to IA-EBL, except that all examples that have ever been seen are saved. After each new batch is processed, the "full memory" version of IA-EBL reads in the full set of examples and then processes it in the same way that it processes a batch of new examples; this allows the learner to correct any classification errors on old examples that were introduced in setting the weights for the new examples.

5.2.4. Interpreting the results

Figure 3 shows the results of the experiments. On the bridge domain, IA-EBL with full memory had exactly the same error rates as IA-EBL with no memory, so only one curve is shown. On both domains, A-EBL's hypotheses are on the average slightly more accurate

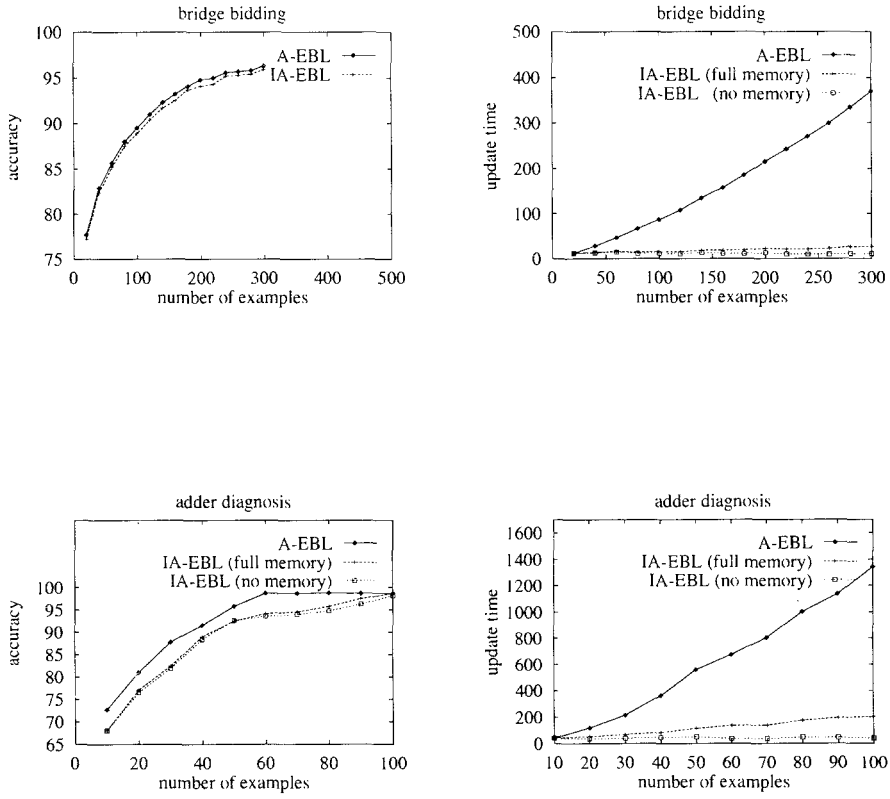

Figure 3. Comparison of A-EBL and IA-EBL.

than IA-EBL's. Although statistically significant,[9] the actual difference in accuracy is small: the average difference between A-EBL and IA-EBL is only 0.6% on the bridge domain and 3.8% on the circuit domain.

However, as figure 3 shows, IA-EBL is much faster in this incremental setting. In the bridge domain, the no-memory version of IA-EBL averages about 11.4 seconds[10] for each update, while A-EBL averages 170.4 seconds. In the circuit domain, IA-EBL averages 42.9 seconds to A-EBL's 625.8 seconds. Thus, in both cases, time is reduced by a factor of about 15. More importantly, while A-EBL's training time grows with the number of examples seen, IA-EBL's training time stays constant: in both domains, the average time for the last five updates is actually slightly *less* than the average time for the first five updates.

Somewhat surprisingly, IA-EBL is faster even when used *non-incrementally*: for both problems, the cumulative training time for IA-EBL was less than half the time required for A-EBL to train on the entire problem set once. This constant-factor speedup in the non-incremental setting is, of course, not scientifically significant. However, it does strongly support our claim that IA-EBL is an efficient incrementalization of A-EBL.

The full-memory version of IA-EBL is also faster than A-EBL, although not as fast as the no-memory version. Unlike the no-memory version of IA-EBL, the training time for the full-memory version of IA-EBL grows (slowly) over time. The full-memory version

of IA-EBL also is less accurate than A-EBL, suggesting that the difference in accuracies is not due simply to the information that has been lost by discarding old examples, but is due instead to the differences in the learning algorithms.

One disadvantage of IA-EBL is that it takes somewhat longer to classify a new example using the hypotheses of IA-EBL than the hypotheses of A-EBL; in the bridge domain, for example, classification averaged 0.32 seconds for A-EBL's hypotheses, and 0.53 seconds for IA-EBL's hypotheses. This is because to classify with A-EBL's hypotheses, it is only necessary to find a single explanation from the selected set, while to classify with IA-EBL's hypotheses, all explanations must be enumerated and their weights retrieved.

5.3. Incrementalizing ANA-EBL

In previous work (Cohen, 1990), we also described an extension to A-EBL called ANA-EBL. In this extension, the set of candidate generalizations generated is not the set of all generalizations formed by applying EBG to some explanation structure of a positive example, but rather the set of generalizations formed by first marking up to k internal nodes of some explanation structure as operational, and then applying EBG to the resulting explanation structure. These extended generalizations can be matched by a new problem that is similar to, but slightly different from, the original training example; in particular, they can be matched by any example with an explanation that differs from the training example in up to k sub-proofs. Inconsistent generalizations are then filtered out as before, and the greedy set cover algorithm is used to find a small set of the remaining candidate rules.

A parallel extension to IA-EBL can be made by simply changing the set of candidate rules $\mathcal{E}_x$. Inspection of the proof shows that the formal analysis still holds for this extended algorithm, which we will call IANA-EBL. Repeating the experiment described above for $k = 1$ shows that the generalization behavior of IANA-EBL is comparable to that of ANA-EBL for the full memory version of IANA-EBL, but somewhat worse for the no-memory version of IANA-EBL (see figure 4). Again, IANA-EBL is much faster, especially the no-memory version.

Notice, however, that on this experiment there is a substantial gap in speed between the full-memory and no-memory versions of IANA-EBL, which was not the case before. Using

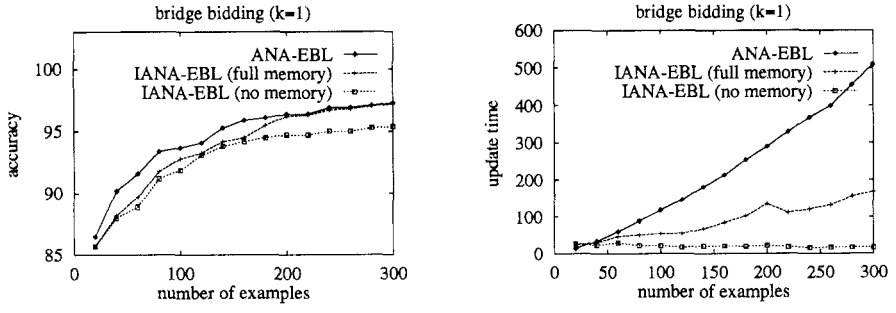

Figure 4. Comparison of ANA-EBL and IANA-EBL.

the full-memory version of IANA-EBL still provides a large speedup with little loss in accuracy; however, the advantage in efficiency gained by using the incremental technique is substantially diminished. The reason for this is not clear. One contributing factor is that while both ANA-EBL and IANA-EBL ultimately rely on enumerating all the explanations of an example, the current implementation of ANA-EBL makes use of special techniques to prune some of the explanations, while IANA-EBL does not. These pruning techniques contribute more when k is large.[11]

5.4. Summary of results

To summarize, formal results indicate that both A-EBL and IA-EBL will produce accurate hypotheses from a polynomial number of examples, and that IA-EBL will be faster in the limit if all the examples are of bounded size. Experimental results on two domains show that IA-EBL provides an order-of-magnitude speedup in time complexity in exchange for a relatively small reduction in accuracy. Finally, additional experiments show that IA-EBL is also an efficient incrementalization of ANA-EBL, at least for small values of k. However, the gains from incrementalization are smaller than was the case for vanilla A-EBL.

6. Related work

In previous work, we argued that A-EBL improves on existing theory-specialization techniques—notably IOE (Flann & Dietterich, 1989) and IVSM (Hirsh, 1990)—in that it simultaneously handles relational initial theories, the multiple explanation problem, and disjunctive target theories. IA-EBL inherits these properties from A-EBL, and is additionally efficient as an incremental learning algorithm. IA-EBL also satisfies a somewhat stronger formal criterion of learnability than A-EBL.

Some recent theory revision systems (Bergadano & Giordana, 1990; Pazzani et al., 1991; Richards & Mooney, 1991; Wogulis, 1991) have capabilities comparable to A-EBL; in addition, these systems can correct several other types of errors in initial theories. However, none of these other systems yet have the formal guarantees on performance that A-EBL or IA-EBL have; also, none of them are incremental. Another class of related systems are those that use abduction to complete an incomplete theory (O'Rorke, 1988; O'Rorke et al., 1989). This work is complementary to A-EBL in that it addresses theory completion (i.e., generalization) rather than theory specialization.

As noted above, the IA-EBL algorithm is closely related to the WINNOW1 algorithm (Littlestone, 1988). Another possible incremental learning algorithm would be simply to use an instantiation of WINNOW1 in which each feature is associated with a *possible* EBG rule, such that the feature is "on" for an example if and only if the rule is applicable. Littlestone's analysis of WINNOW1 suggests that this alternative algorithm would converge in time linear in the size of the target concept and logarithmic in the number of possible EBG rules. Unfortunately, for many theories the number of possible EBG rules is extremely large or (for recursive theories) infinite; in these cases, this approach would be quite inefficient. One can think of the learning algorithm employed in IA-EBL as an extension of

WINNOW1 to infinite sets of attributes; IA-EBL can thus learn even for recursive theories, for which there are infinitely many possible EBG rules. Furthermore, IA-EBL's sample complexity is completely independent of the number of possible EBG rules, and only logarithmically dependent on the size of the initial theory.

The fact that WINNOW1 can be strengthened to deal with an infinite number of potentially relevant attributes is perhaps somewhat surprising. Formal results (Blum et al., 1991) show that, in general, an algorithm (such as WINNOW1) that tolerates exponentially many irrelevant attributes can be converted to an algorithm that tolerates infinitely many irrelevant attributes. However, our mistake bound for IA-EBL is much tighter than the mistake bound of the general construction used in (Blum et al., (1991). In particular, their mistake bound depends on the maximum number of features that are "on" for any example (in the case of IA-EBL, this corresponds to the maximum number of explanations of an example), while ours does not. However, IA-EBL's time complexity does depend on the number of possible explanations for an example.

At a high level, the IA-EBL algorithm can be viewed as using a perceptron learning algorithm to learn a concept using explanations (or more accurately, the applicability of rules derived from explanation structures) as features. IA-EBL thus represents one approach to integrating explanation-based methods with neural network learning methods. Several alternative schemes for integrating these techniques have been proposed. Katz (1989) describes an EBL technique that can be used to reduce the time it takes for a neural network to operate. This corresponds roughly to using EBL to speed up theorem proving in a theory; the neural network corresponds to the logical theory, and the operation of introducing new links between nodes in the network corresponds to adding rules to the logical theory. This integration scheme is quite different from IA-EBL; even the goal of learning is different (Katz's goal is improving performance, rather than improving classification accuracy.) Towell et al. (1990) describe a technique called KBANN, in which an explanation structure derived from a propositional domain theory is used to choose an initial network topology for a neural network. The network is then trained further using the backpropagation learning algorithm. This technique is more similar to IA-EBL; however, in IA-EBL, explanations are used as features to be fed into a network, while in KBANN, explanations are used to form the network itself. Unlike IA-EBL, KBANN is applicable only to propositional domain theories; however, also unlike IA-EBL, it can make use of theories that are incomplete.

7. Conclusions

This article has described an alternative version of the A-EBL learning system called IA-EBL that addresses the problem of learning incrementally. IA-EBL can be viewed as using a variant of the perceptron learning algorithm to learn a concept described by a set of EBG rules. This is a novel way of *using* EBG rules: the learning algorithm ascribes a weight to each rule, and a new instance x is assumed to be a member of the learned concept only if the sum of the weights of the explanations relevant to x is above a certain threshold. The main disadvantage of IA-EBL relative to A-EBL is that IA-EBL's hypotheses are not expressed as Horn clause theories but rather as a weighted sum over a set of rules. This

means that the hypotheses of the system are somewhat harder for people to understand. It is also generally the case that it is slightly less efficient to use the hypotheses of the incremental system to classify novel instances. Unlike A-EBL, however, IA-EBL is an efficient incremental learner.

Formally, the IA-EBL algorithm was shown to be efficient according to Littlestone's mistake-bounded model of incremental learning. A consequence of this is that the predictions made by IA-EBL become probably approximately correct after a polynomial number of training examples. It was also shown analytically that IA-EBL converges to any target theory that is learnable using A-EBL.

Experimentally, IA-EBL was shown to be efficient in incremental settings, in both time and sample complexity. In time complexity, it was shown to be far superior to A-EBL when used in an incremental setting, providing an order-of-magnitude speedup in two different test domains. Finally, an incremental version of ANA-EBL, an extension of A-EBL, was described, and shown to improve on ANA-EBL in efficiency, although not as dramatically as IA-EBL improves on A-EBL.

Acknowledgments

The author is grateful to Haym Hirsh and Russell Greiner for their comments on a draft of this article, to Susan Cohen for proofreading it, and to Paul Utgoff and three anonymous *Machine Learning* reviewers for many helpful comments. Russell Greiner also suggested the circuit diagnosis problem, and performed some initial experiments indicating that it would be an interesting application of A-EBL.

Notes

1. We use $\mathfrak{X}_S$ to denote the characteristic function of a set S; recall that the characteristic function $\mathfrak{X}_S(x)$ of a set S is 1 if $x \in S$ and 0 otherwise.
2. In Littlestone's original model, this information was encoded as a *reinforcement*, which indicated whether or not the prediction was correct.
3. This is the size measure used in A-EBL. It was chosen for A-EBL because it is easy to analyze, and because it allowed some important optimizations in the implementation. We retain the same size measure here to lessen the differences with A-EBL.
4. A set S of strings is *prefix free* if, whenever a string s is a member of S, no prefix of s is a member of S.
5. For example, one might assume that component 1 is faulty in one part of the specialized theory, and that component 1 is functioning normally in another part. Abductive reasoning systems would discard such inconsistent hypotheses.
6. There are 23 levels of logic involved in computing the final bit of the sum.
7. The property of needing to examine a datum many times before it is properly treated is inherited from the WINNOW1 perceptron learning algorithm and is common to many neurophysiologically inspired learning algorithms.
8. Notice that using an ordering procedure is perfectly compatible with the formal results, which hold for any ordering, even an adversarial one.
9. All apparent differences in the graphs in this article are statistically significant at the 99.5% level or above. All differences other than the difference in accuracy between IA-EBL with full memory and no memory on the circuit domain are statistically significant at the 99.999% level or above.

10. Time is measured in CPU seconds on a Sparc 1+. A-EBL and IA-EBL are both implemented in Quintus Prolog 3.1.
11. It is even possible that as k is increased, the computational advantage of using IANA-EBL will disappear, at least using the current implementation. Unfortunately, limitations on computer memory made experimental confirmation of this impossible.

References

Abramson, Norman. (1963). *Information theory and coding*. New York: McGraw Hill.

Ali, Karmal. (1989). Augmenting domain theory for explanation-based generalization. In *Proceedings of the Sixth International Workshop on Machine Learning*. Ithaca, NY: Morgan Kaufmann.

Bergadano, Francesco, & Giordana, Attilio. (1990). Guiding induction with domain theories. In *Machine learning: An artificial intelligence approach*, Vol. 3 (pp. 474–492). Morgan Kaufmann.

Blum, Avrim, Hellerstein, Lisa, & Littlestone, Nick. (1991). Learning in the presence of finitely or infinitely many irrelevant attributes. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*. Santa Cruz, CA: Morgan Kaufmann.

Blum, Avrim. (1990). Separating PAC and mistake-bound learning models over the boolean domain. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*. Rochester, NY: Morgan Kaufmann.

Cohen, William W. (1990). Learning from textbook knowledge: A case study. In *Proceedings of the Eighth National Conference on Artificial Intelligence*. Boston, MA: MIT Press.

Cohen, William W. (1991). The generality of overgenerality. In *Proceedings of the Eighth International Workshop on Machine Learning*. Ithaca, NY: Morgan Kaufmann.

Cohen, William W. (1992). Abductive explanation based learning: A solution to the multiple inconsistent explanation problem. *Machine Learning, 8(2)*, 167–219.

Cohen, William W. (1992). Compiling knowledge into an explicit bias. In *Proceedings of the Ninth International Conference on Machine Learning*. Aberdeen, Scotland: Morgan Kaufmann.

Danyluk, Andrea. (1989). Finding new rules for incomplete theories. In *Proceedings of the Sixth International Workshop on Machine Learning*. Ithaca, NY: Morgan Kaufmann.

de Kleer, Johan, & Williams, Brian C. (1987). Diagnosing multiple faults. *Artificial Intelligence, 32*, 97–130.

DeJong, Gerald, & Mooney, Raymond (1986). Explanation-based learning: An alternative view. *Machine Learning, 1(2)*, 145–176.

Drastal, George, Czako, Gabor, & Raatz, Stan. (1989). Induction in abstraction spaces: A form of constructive induction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*. Detroit, MI: Morgan Kaufmann.

Fawcett, Tom. (1989). Learning from plausible explanations. In *Proceedings of the Sixth International Workshop on Machine Learning*. Ithaca, NY: Morgan Kaufmann.

Flann, Nicholas & Dietterich, Thomas. (1989). A study of explanation-based methods for inductive learning. *Machine Learning, 4(2)*, 187–226.

Hirsh, Haym. (1989). Combining empirical and analytic learning with version spaces. In *Proceedings of the Sixth International Workshop on Machine Learning*. Ithaca, NY: Morgan Kaufmann.

Hirsh, Haym. (1990). *Incremental version space merging: A general framework for concept learning*. Boston, MA: Kluwer Academic Publishers.

Katz, Bruce F. (1989). Integrated learning in a neural network. In *Proceedings of the Sixth International Workshop on Machine Learning*. Ithaca, NY: Morgan Kaufmann.

Kearns, Micheal, Li, Ming, Pitt, Leonard, & Valiant, Les. (1987). On the learnability of boolean formulae. In *19th Annual Symposium on the Theory of Computing*. ACM Press.

Langley, Pat, Gennari, John, & Iba, Wayne. (1987). Hill-climbing theories of learning. In *Proceedings of the Fourth International Workshop on Machine Learning*. Irvine, CA: Morgan Kaufmann.

Littlestone, Nick. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning, 2(4)*, 161–186.

Littlestone, Nick. (1989). From on-line to batch learning. In *Proceedings of the 1989 Workshop on Computational Learning Theory* (pp. 269–284). Santa Cruz, CA: Morgan Kaufmann.

Littlestone, Nick. (1989). Mistake bounds and logarithmic linear-threshold learning algorithms. (Technical Report UCSC-CRL-89-11). Department of Computer and Information Sciences, University of California/Santa Cruz.

Mahadevan, Sridhar. (1989). Using determinations in EBL: A solution to the incomplete theory problem. In *Proceedings of the Sixth International Workshop on Machine Learning.* Ithaca, NY: Morgan Kaufmann.

Mooney, Ray, & Ourston, Dirk. (1989). Induction over the unexplained. In *Proceedings of the Sixth International Workshop on Machine Learning.* Ithaca, NY: Morgan Kaufmann.

O'Rorke, Paul, Morris, Steven, & Schulenburg, David. (1989). Theory formation by abduction: Initial results of a case study based on the chemical revolution. In *Proceedings of the Sixth International Workshop on Machine Learning.* Ithaca, NY: Morgan Kaufmann.

O'Rorke, Paul. (1988). Automated abduction and machine learning. In *Proceedings of the 1988 Spring Symposium on Explanation Based Learning.* Palo Alto, CA: AAAI.

Ourston, Dirk, & Mooney, Raymond. (1990). Changing the rules: A comprehensive approach to theory refinement. In *Proceedings of the Eighth National Conference on Artificial Intelligence.* Boston, MA: MIT Press.

Pazzani, Michael, Brunk, Clifford, & Silverstein, Glenn. (1991). A knowledge-intensive approach to learning relational concepts. In *Proceedings of the Eighth International Workshop on Machine Learning.* Ithaca, NY: Morgan Kaufmann.

Pazzani, Michael. (1988). Selecting the best explanation in explanation-based learning. In *Proceedings of the 1988 Spring Symposium on Explanation Based Learning.* Palo Alto, CA: AAAI.

Rajamoney, Shankar, & DeJong, Gerald. (1988). Active explanation reduction: An approach to the multiple explanation problem. In *Proceedings of the Fifth International Conference on Machine Learning.* Ann Arbor, MI: Morgan Kaufmann.

Reiter, Ray. (1987). A theory of diagnosis from first principles. *Artificial Intelligence, 32,* 57–95.

Richards, Bradley, & Mooney, Raymond. (1991). First-order theory revision. In *Proceedings of the Eighth International Workshop on Machine Learning.* Ithaca, NY: Morgan Kaufmann.

Rosenbloom, Paul, & Aasman, Jans. (1990). Knowledge level and inductive uses of chunking (ebl). In *Proceedings of the Eighth National Conference on Artificial Intelligence.* Boston, MA: MIT Press.

Towell, Geoffrey, Shavlik, Jude, & Noordewier, Michiel. (1990). Refinement of approximate domain theories by knowledge-based artificial neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence.* Boston, MA: MIT Press.

Utgoff, Paul. (1989). Incremental induction of decision trees. *Machine Learning, 4(2).*

Valiant, L.G. (1984). A theory of the learnable. *Communications of the ACM, 27(11),* 1134–1142.

Wogulis, James. (1991). Revising relational domain theories. In *Proceedings of the Eighth International Workshop on Machine Learning.* Ithaca, NY: Morgan Kaufmann.

Received May 13, 1992
Accepted September 28, 1992
Final Manuscript February 8, 1993