Learning by Discrimination: A Constructive Incremental Approach

Christophe Giraud-Carrier and Tony Martinez
Department of Computer Science, Brigham Young University, Provo, UT, USA
Email: {cgc,martinez}@cs.byu.edu

Abstract— **This paper presents i-AA1*, a constructive, incremental learning algorithm for a special class of weightless, self-organizing networks. In i-AA1*, learning consists of adapting the nodes' functions and the network's overall topology as each new training pattern is presented. Provided the training data is consistent, computational complexity is low and prior factual knowledge may be used to "prime" the network and improve its predictive accuracy and/or efficiency. Empirical generalization results on both toy problems and more realistic tasks demonstrate promise.**

Index Terms— **Incremental learning, Self-organizing networks, Constructive learning, Prior knowledge**

I. Introduction

Motivated by an interest in biological plausibility, connectionism offers an alternative to symbolic learning in classical AI. Connectionist models are networks consisting of large numbers of computationally simple, highly interconnected nodes. The pattern of (generally) weighted connections between nodes defines the network's topology. In most cases, a network's topology is static and learning is effected by modifying the connections' weights as the system iterates over a set of training patterns in an attempt to minimize misclassification error. When it remains static, a network's topology becomes one of its strongest source of learning bias. Unfortunately, this architectural bias has also proven to be one of the most difficult to use effectively in practice. Despite numerous attempts at discovering both theoretical and empirical rules for the selection of a best *a priori* topology for learning a particular task, the actual choice of a topology remains a fairly *ad hoc* procedure [20], [29], [36], [44].

Over the past decade however, many researchers have eluded this difficult issue by focusing their efforts on the design of constructive learning methods. In addition to modifying the connections' weights, constructive algorithms also modify the network's topology. Hence, the topology is no longer a user-defined, hand-crafted parameter of the learning process. Instead, the network grows in a self-organizing fashion to best fit the problem. In other words, the dynamic addition and deletion of nodes in the network become intrinsic parts of learning and the network's topology emerges naturally as a result of these transformations. As parsimony is often a concern, constructive procedures may use the general Occam's Razor Principle as a bias to favor smaller networks. The constructive approach thus provides the learner with added flexibility. Empirical studies have also demonstrated that constructive algorithms yield generalization results that compare favorably with those of their static counterparts.

Adaptive Self-Organizing Concurrent Systems (ASOCS) define a special class of weightless, dynamic networks [23]. ASOCS networks learn by adapting their nodes' functions and by modifying their overall topology. Once a network's inputs are set, the network computes its output as data flow asynchronously and in parallel through the network. When learning, the training instances or patterns are presented one at a time over time. The network executes first and compares its computed output with the target output. If target and computed outputs are different, the network adapts to learn the new pattern, through structural and functional changes. If the outputs are identical, only minor changes are made to facilitate later adaptation. ASOCS' target applications are binary classification tasks for Boolean-valued patterns. Several learning algorithms have been developed for ASOCS, including Adaptive Algorithm (AA)1 [24], AA2 [26] and AA3 [27]. Although all three algorithms have been shown to converge on any arbitrary Boolean pattern set, only AA1 exhibits meaningful off-training set generalization [4], [13].

AA1 learns by discrimination and constructs a network in which knowledge is distributed across all of the nodes. In a previous paper, we presented AA1*, an extension of AA1 that focuses on minimizing the network's growth to improve predictive accuracy and making lower-order features available for discrimination during Node Selection [14]. Here, we formalize and fill in many of the details missing from AA1*'s earlier account. In addition, we make an explicit assumption of consistency, thus enabling true incremental learning [16], and demonstrate the algorithm's ability to exploit prior factual knowledge in learning. This new, incremental version of AA1*, called i-AA1*, requires less memory and is more computationally efficient than both of its predecessors.

The paper is organized as follows. Section II outlines related work. Sections III and IV provide an overview of i-AA1*'s underlying architecture and learning paradigm. Section V reports experimental results on several syn-

This paper is an expanded and enhanced version of ''A Constructive Incremental Learning Algorithm for Binary Classification Tasks,'' by C. Giraud-Carrier and T. Martinez, which appeared in the Proceedings of the IEEE Mountain Workshop on Adaptive and Learning Systems, Logan, UT, USA, July 2006, pp. 213-218. © 2006 IEEE.

thetic datasets and more realistic applications. Section VI shows how prior knowledge is naturally incorporated in i-AA1*'s learning and presents experimental results that demonstrate the utility of prior knowledge in increasing parsimony and predictive accuracy. Finally, section VII concludes the paper.

II. Related Work

There has been considerable work in designing constructive learning algorithms, mainly within the context of neural networks, such as Target Switch [5], Cascade-Correlation [8], Upstart [10], Offset [22], MOST [3], and many others (e.g., see [11], [21], [30], [33]). For the record, it is interesting to note here that, although it became most popular in the 1990s, constructive learning is by no means a new idea. Indeed, evidence suggests that a form of constructive learning was investigated by Alan Turing as early as 1948 in his B-type unorganized machine [6].

Most constructive approaches are based on feedforward neural networks and tend to restrict the constructive mechanisms to the design of the network's topology. Our proposed algorithm acts on ASOCS, a class of weightless logic networks, and effects learning by modifications to both the network's topology and the nodes' functions. Incidentally, these are exactly the kinds of adaptive mechanisms used in Turing's B-type machine. As far as network construction, our approach is closest to that of the Target Switch algorithm [5]. Indeed, i-AA1*'s one-sided discriminant nodes essentially induce $\oplus$- and $\ominus$-dichotomies and similarly serve as the main building blocks in the network's construction. However, unlike i-AA1*, the Target Switch algorithm supports analog inputs but it is non-incremental.

The choice of incremental learning in i-AA1* is motivated by arguments put forth in, for example [7], [16], on the ubiquity of incremental tasks. Although work incremental learning has been somewhat sporadic, there are some notable exceptions in the areas of decision tree learning [43], unsupervised learning [9], and neural network learning [35].

Since it implements symbolic knowledge (i.e., simple logical implications) on a neural network structure, i-AA1* can be viewed as a special instance of the class of hybrid symbolic connectionist learning systems. An excellent survey of such systems is in [28]. We mention only the most relevant ones here. ScNets aim at providing an alternative to knowledge acquisition from experts [18]. Known rules may be pre-encoded and new rules can be learned inductively from examples. The representation lends itself to rule generation but the constructed networks are complex and generalization does not appear trivial. CONSYDERR is strictly concerned with a connectionist approach to concept representation and commonsense reasoning, and does not address the problem of learning (how such a skill could be incorporated is also unclear) [38], [39]. KBANN expresses prior knowledge identically to i-AA1* as pre-encoded precepts [41], [42]. However, unlike i-AA1*, KBANN translates the given precepts into an equivalent artificial neural network, which it may then perturb and learn using the standard backpropagation algorithm. Extensions of KBANN have been proposed, where the topology is also modified during learning through constructive mechanisms [34]. Other hybrid approaches have used symbolic knowledge in the form of fuzzy rules to derive a regularization term for neural network learning [19].

The philosophy and motivation underlying precepts in i-AA1* are the same as those of (independently proposed) hints [1], [2]. The notion of a hint is more general than that of a precept, allowing such information as symmetry or invariance properties to be captured. Unlike precepts that are explicitly encoded as rules and learned as such by i-AA1*, hints are represented by virtual examples which must be learned by the network alongside the training set, either directly as part of the learning task or as a catalyst to the target learning task [37].

III. i-AA1*: Discrimination

In this section, we show how training patterns are represented and generalized. We present i-AA1*'s architectural constraints, show how the network implements knowledge in a distributed fashion using the notion of discrimination.

A. Knowledge Representation

The target applications of i-AA1* are binary classifications of Boolean-valued patterns. A training pattern consists of a set of values (typically one for each Boolean input) together with a target class value. By convention, patterns of class 0 are called *negative* patterns, and patterns of class 1 are called *positive* patterns. Furthermore, patterns of the same class are said to be *concordant* with respect to each other, whilst patterns of opposite classes are *discordant*. Figure 1 shows three sample patterns in the case of three input variables A, B and C. The target class is labeled Z and we use the standard Boolean notation: A means that variable A has value 1 and $\overline{A}$ means variable A has value 0.

$$\begin{array}{lcl} A\,\overline{B}\,\overline{C} & \rightarrow & Z \\ \overline{A}\,B\,\overline{C} & \rightarrow & \overline{Z} \\ A\,B\,C & \rightarrow & Z \end{array}$$

Figure 1. Sample Patterns with 3 Inputs

In Figure 1, the first pattern corresponds to variable A having value 1, and variables B and C having value 0. The first and third patterns are positive. The second pattern is negative. Hence, the first and third patterns are concordant, whilst the first and second patterns, as well as the second and third patterns, are discordant.

An important aspect of i-AA1* (and all other ASOCS algorithms) is that input values may be omitted from training patterns. The semantics associated with variables whose value is omitted are different from those associated with variables whose value is unknown. If p is a pattern in which the value of input variable V is omitted, then

p represents both patterns that would be obtained from p by adding V and $\overline{V}$, respectively, to p. For example, with the three inputs A, B and C, the pattern $A\,C \rightarrow Z$ represents both patterns $A\,B\,C \rightarrow Z$ and $A\,\overline{B}\,C \rightarrow Z$. In other words, $A\,C \rightarrow Z$ means that when A and C have value 1, then Z has value 1, regardless of the values of B. It follows that omitting the value of a variable may be viewed as adding information to the system's knowledge. On the other hand, variables whose value is unknown convey a lack of information, often attributable to the fact that patterns extracted from real-world applications may be incomplete.[1]

For historical reasons, we refer to training patterns with variables whose values are omitted as precepts [12], [15]. By using prior knowledge in the form of precepts together with raw patterns, i-AA1* can effectively combine the intensional approach (based on features, expressed here by precepts) and the extensional approach (based on instances, expressed here by patterns) to learning. It is clear that this combination increases flexibility. On the one hand, extensionality accounts for the system's ability to adapt to its environment, i.e., to be more autonomous. On the other hand, intensionality provides a mechanism through which the system can be taught and thus does not have to unnecessarily suffer from poor or atypical learning environments. Note that this ability to "teach" the network is non-existent in most extant neural network systems. Furthermore, whereas traditional neural networks are opaque and thus require complex rule extraction mechanisms, the knowledge encoded in an i-AA1* network can readily be transformed back into its logical equivalent form and minimized using existing tools.

B. Knowledge Implementation

We begin by stating an essential assumption of i-AA1* for correct knowledge implementation. As in standard logic, we say that a set of patterns is inconsistent if there exists an assignment of values for all of the input variables that makes the premises of two discordant patterns true simultaneously.

At the cost of storing all training patterns and significant computational overhead, AA1 and AA1* implement a simple, chronology-based method to handle inconsistencies online [13], [25]. By contrast, i-AA1* assumes that the set of training patterns is consistent (i.e., noise-free). Although somewhat restrictive, this assumption results in significant memory and computational savings.

Architecturally, the structure and connectivity of an i-AA1* network differ significantly from those of classical connectionist models. The overall network structure is a rooted graph in which all computations are effected by simple broadcast-and-gather mechanisms. The network's inputs are located at leaf nodes and the network's output is the output of its root or top node. Nodes in i-AA1* have the following characteristics.

- Each node has exactly two inputs and one output.
- A node's function is either AND or OR, extended in a natural way to accommodate missing input values denoted by ? (e.g., 0 AND ? is 0, 0 OR ? is ?).
- A node's output is the value of its function applied to its (possibly inverted) inputs.
- A node's output (except for the top node) is sent to one or more other nodes' inputs on weightless connections.
- Each node has a store of local memory, called a node table (NT), in which it records the output it produces for each training pattern.

A node table has two columns, P and N, holding the node's output values for positive and negative patterns, respectively. Figure 2 shows a sample node table.

P	N
0	0
1	?
1	

Figure 2. Sample Node Table

There is, of course, a consistent one-to-one correspondence among the cells of all NTs. Note that the NTs are ignored during execution, where the network functions exclusively as a parallel logic circuit. During learning, the NTs are used only to control discrimination as described in the following section.

C. Discrimination

The notion of discrimination is central to i-AA1* as it governs the learning process. Discrimination takes place at the node level and is defined as follows. Let p be any pattern and K be a node whose output is 0 (respectively, 1) when p is presented to the network. Then K discriminates p from all discordant patterns for which K outputs 1 (respectively, 0).

Node tables provide a straightforward way to assess a node's discrimination capacity. If a cell in a node's NT contains the value 0 or 1, then that node discriminates the corresponding pattern from all patterns of the opposite class whose cells contain the opposite value. For example, assume that the NT of Figure 2 belongs to some node K and consider the first cell of the N column. Let n be the corresponding pattern. If K outputs 0, then it is impossible to decide which of n or the pattern corresponding to the first cell of the P column has been matched. However, if K outputs 1, then it is clear that n is not matched. Now, since K outputs 1 for the patterns corresponding to the second and third cells of the P column, it follows that K discriminates n from these two patterns. Clearly, cells containing ? cannot participate in discrimination.

A node's ability to discriminate a given pattern is captured by its relative discriminant count. Let p be any

[1] i-AA1* does not support the semantics of unknown values. It treats both unknown and omitted values as omitted, which in the former case may lead to artificial inconsistencies.

pattern, S be a set of discordant patterns and K be a node. Then, the relative discriminant count of K with respect to p and S, denoted by $DC_K(p, S)$, is the number of patterns in S that K discriminates from p. For example, if S consists of the three positive patterns corresponding to the P column of the NT of Figure 2 and n is the pattern corresponding to the first cell of the N column, then $DC_K(n, S) = 2$. The relative discriminant count of a set of nodes is simply the sum of the relative discriminant counts of its members.

A node's ability to discriminate among previously seen training instances is captured by its discriminant power. Let K be a node, and N_i and P_i ($i = 0, 1$) be the numbers of cells in K's node table containing the value i. The discriminant power of K, denoted by DP_K, is given by $DP_K = N_0P_1 + N_1P_0$. The discriminant power of a node is thus a global measure of how well that node discriminates past training instances.

Two special kinds of nodes are defined here as they are central to i-AA1*'s learning.

1) A node that discriminates every positive pattern from every negative pattern is a *complete discriminant* node. A complete discriminant node outputs one value (0 or 1) for all positive patterns and the opposite value for all negative patterns.
2) A node that discriminates at least one pattern from all patterns of the opposite class is a *one-sided discriminant* node. A one-sided discriminant node outputs the same value for either all positive or all negative patterns and the opposite value for at least one discordant pattern.

An i-AA1* network converges on a training set if, after presentation of the training set, the network's top node is complete discriminant (with values 1 in the P column). One-sided discriminant nodes, like the $\oplus$- and $\ominus$-dichotomies of [5], provide useful building blocks. Intuitively, it seems reasonable to devise an algorithm wherein successive constructions and connections of one-sided discriminant nodes incrementally lead to a network whose top node is complete discriminant (with values 1 in the P column). This intuition, which is formalized in the following section, is the basis of i-AA1*'s learning algorithm.

IV. i-AA1*: Learning Algorithm

As mentioned earlier, the target applications of i-AA1* are binary classification tasks and learning is incremental. Let $p_1, p_2, \ldots, p_m$ be a sequence of training patterns, presented one at a time. Then, i-AA1* produces a sequence $iNet_1, iNet_2, \ldots, iNet_m$ of networks such that:

1) $iNet_{k+1}$ depends only on $iNet_k$ and p_{k+1}, for all $1 \leq k \leq m-1$.
2) The top node of $iNet_k$ is complete discriminant, for all $1 \leq k \leq m$.

By convention, $iNet_1$ is a degenerate network whose output is set to the value of the output of p_1. The learning algorithm for i-AA1* is shown in Figure 3. We assume that $m > 1$.

1) **Initialization**
   a) Construct input nodes for all input variables
   b) Set the network's output to the output of p_1
2) **Incremental Learning**
   - For $k = 1$ to m
     a) Execute the network on p_k with NT update
     b) If (network's output $\neq p_k$'s output value)
        – Perform node selection
        – Perform node combination

Figure 3. i-AA1* Learning Algorithm.

Step 1 is executed only once to initialize the network. So long as they remain unconnected, input nodes do not really form part of the network. The (degenerate) network's output is initialized to the value of the output of the first training pattern. The first non-degenerate network is created when the first discordant instance is encountered.

In step 2, i-AA1* learns incrementally to discriminate each new training pattern from all previously discordant patterns. The training pattern is presented to the network for execution, which includes updating the NTs so that each node's NT stores that node's output for the new pattern. If the network's output is the same as the training pattern's target class value, then no changes need be made. Otherwise, adjustments must be made to the network to create a new complete discriminant top node. Note that since the training set is not stored explicitly, the NTs of unconnected input nodes must be updated, otherwise it would be impossible to construct the first one-sided discriminant node to discriminate the first discordant pattern from all previous patterns. Furthermore, updating the NTs of unconnected input nodes makes these nodes' discriminating capacities available for use in the constructive mechanisms of step 2b.

Given that, by construction, the current top node is complete discriminant for all patterns except the incoming one, the procedure followed by i-AA1* to adjust the network consists of:

1) Constructing a one-sided discriminant node that discriminates the incoming pattern from all discordant patterns.
2) Combining this one-sided discriminant node with the current top node to obtain the new desired complete discriminant top node.

The construction of an appropriate one-sided discriminant node involves recruiting/selecting discriminant nodes in the network and combining them. The following theorem provides the motivation for this construction, whilst the assumption of consistency for the training set guarantees its correctness, i.e., the selection process will terminate with a set of nodes that together discriminate the incoming training pattern from all previous discordant patterns.

Theorem 1: (adapted from Lemma 2 of [13]) Let p be the incoming training pattern and D the corresponding set of discordant patterns. If N_1 and N_2 are nodes such that N_1 discriminates p from $D_1 \subseteq D$ and N_2 discriminates p from $D_2 \subseteq D$, then the parent node resulting from

combining N_1 and N_2 discriminates p from $D_1 \cup D_2$. This result extends naturally to show that, given an appropriate set of discriminant nodes, a new network whose top node is complete discriminant, can be constructed [13]. The details of node selection and node combination are given in the following sections.

Note that, by construction, i-AA1* satisfies the following form of (strong) convergence.

Theorem 2: Let $p_1, \ldots, p_m$ be a sequence of training patterns and $iNet_i$ be the i-AA1* network following presentation of p_i. Then, N_i gives the correct output for all patterns p_j, $1 \leq j \leq i$.

A. Node Selection

Node selection consists of recruiting nodes, both from the network and from the set of yet unconnected input nodes, according to their ability to discriminate the incoming pattern from discordant patterns. Figure 4 shows how node selection takes place.

1) **Initialization:** Let p be the incoming pattern, D be the set of all discordant patterns p must be discriminated from and S be the set of all available nodes. For each node N_i, let D_i denote the set of patterns in D that N_i discriminates from p.
2) **Optimization loop:** Set $k = 1$. Repeat until $k > num\_iterations$
   a) Set $S_k = \emptyset$ and $i = 1$.
   b) **Main loop:** Repeat until $\max_{N \in S}\{DC_N(p, D - \cup_{j=1}^{i-1} D_j)\} = 0$
      i) Let $W = \{M \in S : DC_M(p, D - \cup_{j=1}^{i-1} D_j) = \max_{N \in S}\{DC_N(p, D - \cup_{j=1}^{i-1} D_j)\}$.
      ii) Select $N_i \in W$ s.t. $DP_{N_i} = \max_{N \in W}\{DP_N\}$.
      iii) Add N_i to S_k.
      iv) $i = i + 1$.
   c) Remove N_1 from S.
   d) $k = k + 1$.
3) **Minimal set selection:** Select S_j s.t. S_j is the smallest of the S_i's with maximal relative discriminant count.

Figure 4. Node Selection.

The main loop successively recruits the node that discriminates the incoming pattern p from the largest number of remaining, non-discriminated, discordant patterns. That is, the node to be selected next has the largest relative discriminant count of all the nodes that have not been selected previously (step 2(b)i). Ties are broken using discriminant power, namely by selecting the node with the largest discriminant power (step 2(b)ii). This greedy selection process goes on until all the nodes that have not yet been selected have relative discriminant count 0. The main loop is guaranteed to terminate. Because the search through the network is essentially best-first, the set, S_1, of selected nodes produced by executing the main loop exactly once satisfies the following properties:

Property 1: $DC_{S_1}(p, D) = |D|$.

Property 2: S_1 is not minimal.

The first property follows from Theorem 1 and the assumption of consistency. It shows that S_1 has maximal relative discriminant count. The second property shows, however, that S_1 may not do so optimally in the sense that there may exist smaller sets of nodes with identical relative discriminating ability. Consider, for example, a set of 5 nodes, $N_1, N_2, \ldots, N_5$. Let p be the training pattern and let $d_1, d_2, \ldots, d_8$ be the discordant patterns from which p must be discriminated. Let Table 5 show which pattern each node discriminates from p in tabular form.

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8
N_1	$\star$	$\star$	$\star$	$\star$	$\star$			
N_2				$\star$	$\star$	$\star$		
N_3					$\star$		$\star$	
N_4	$\star$	$\star$						$\star$
N_5			$\star$	$\star$	$\star$	$\star$		

Figure 5. Sample Discrimination Table

Executing the main loop only once would produce the set $S_1 = \{N_1, N_2, N_3, N_4\}$ or the set $S_2 = \{N_1, N_3, N_4, N_5\}$, due to the non-deterministic selection in step 2(b)(i). Both S_1 and S_2 contain 4 nodes. However, a quick examination of the table in Figure 5 shows that the set $U = \{N_3, N_4, N_5\}$ also discriminates p from all discordant patterns and contains only 3 nodes. We observe the following about U:

Observation 1: N_1 is the first node selected in the execution of the main loop (to construct S_1 or S_2). Assume that N_1 is ignored. Then, execution of the main loop produces the smaller set U.

It is easy to see that optimal node selection is an instance of the MINIMAL COVER problem, which is known to be NP-hard. However, properties 1 and 2, together with the above observation, provide the motivation for the following heuristic generalization. Rather than being executed only once, the main loop is embedded within an optimization loop. In the optimization loop, the main loop is executed $num\_iterations$ times, each time leaving out the nodes that first won the competition for largest relative discriminant count (step 2(b)(i)) in the previous executions of the main loop. Because of Property 1, further iterations through the optimization loop are strictly aimed at reducing the number of nodes selected. The optimization loop clearly improves the chances of finding a smaller set of discriminant nodes and thus of constructing smaller networks. In the current implementation, $num\_iterations$ is set to 5 as empirical studies suggest that, in most cases, the smallest set is achieved after 2 or 3 iterations only. Because $num\_iterations$ is fixed (and small), the optimization loop increases learning time by only a small constant multiplicative factor. Execution time may be significantly reduced, however, since smaller networks are constructed.

At the conclusion of the optimization loop, the smallest set of selected nodes with maximal relative discriminant count is chosen. Given the non-deterministic selection, it is not possible to simply check subsets. For example, U is a subset of S_2, but not of S_1.

B. Node Combination

Once a sufficient set of nodes has been obtained by selection, the nodes are combined in pairs as shown in Figure 6.

1) **Initialization:** Let S be the set of discriminant nodes resulting from node selection. Let CTN denote the current network's top node.
2) **One-sided discriminant node construction:** While $|S| > 1$
   a) Select 2 nodes, N_1 and N_2, from S.
   b) Combine N_1 and N_2 to create parent node P.
   c) Remove N_1 and N_2 from S.
   d) Add P to S.
3) **New top node construction:** Let $OSDN$ be the node in S. Combine $OSDN$ and CTN to create parent node NTN.

Figure 6. Node Combination

By Theorem 1, $OSDN$ is a one-sided discriminant node that discriminates the incoming pattern from all discordant patterns and NTN is the expected new complete discriminant top node of the network. Note that if CTN does not exist, $OSDN$ (step 3) becomes NTN. A brief description of how parent nodes are initialized follows.

The function of all non top-node parent nodes P is set based on the values produced by their children on the incoming pattern p so that the output of the parent for p is 0. Formally, let L be the node to be used as left input and R be the node to be used as right input to P. Let L_p and R_p be the values output by L and R, respectively, when the incoming pattern is presented to the network. P's function f is set as described in Figure 7.

Case

$L_p = 0 \wedge R_p = 0$: $f = (L \text{ OR } R)$
$L_p = 0 \wedge R_p = 1$: $f = (L \text{ OR } \overline{R})$
$L_p = 1 \wedge R_p = 0$: $f = (\overline{L} \text{ OR } R)$
$L_p = 1 \wedge R_p = 1$: $f = (\overline{L} \text{ OR } \overline{R})$

Figure 7. Parent Node Function Setting

The setting of NTN's function is slightly different as it must also guarantee that NTN outputs 1 for positive patterns and 0 for negative patterns. Hence, the decision as to which function it must be set to depends on $OSDN$ and the incoming pattern's target class value. One way to set NTN's function is shown in Figure 8, where p_t is the target value of the incoming pattern and $OSDN_p$ is the value output by $OSDN$ when the incoming pattern is presented to the network.

Case

$OSDN_p = 0 \wedge p_t = 0$: $f = (OSDN \text{ AND } CTN)$
$OSDN_p = 0 \wedge p_t = 1$: $f = (\overline{OSDN} \text{ OR } CTN)$
$OSDN_p = 1 \wedge p_t = 0$: $f = (\overline{OSDN} \text{ AND } CTN)$
$OSDN_p = 1 \wedge p_t = 1$: $f = (OSDN \text{ OR } CTN)$

Figure 8. Top Node Function Setting

Once a parent node's function has been set, it is straightforward to fill in its node table. This is accomplished simply by applying the parent's node function pairwise to the values in corresponding cells of its children's node tables.

Note that in the current implementation, nodes are combined in pairs in random order. Whatever the ordering, the number of combinations is the same, namely exactly $n - 1$ combinations for n discriminant nodes. However, the ordering may affect the depth of the network.

V. Experimental Results

This section reports some experimental results using i-AA1* learning on both synthetic and more realistic problems. The synthetic problems validate the system while the more realistic tasks demonstrate its applicability. The synthetic problems consist of the mirror symmetry problem, the shift detection problem and the three MONK's problems [40]. The other tasks are drawn from the UCI repository [31].

Since i-AA1* handles only Boolean inputs, a few remarks about the encoding of non-Boolean (i.e., nominal/discrete and real-valued) inputs are in order. To be effective, this encoding should provide a natural extension to Boolean discrimination between 0 and 1. For nominal values, it must be such that i-AA1* can discriminate among the *N* values of a given input, where $N > 2$. This is easily accomplished using a feature-based encoding where each nominal input is encoded as *N* binary inputs, such that exactly one binary input is set to 1 for each value of the nominal input. Such an encoding results in the creation of an input node for each attribute-value pair. Hence, for any training instance, i-AA1* can identify which value of each input that instance takes, and subsequently use that information for discrimination. Clearly, node selection and node combination allow the creation of arbitrary combinations of such attribute-value pairs, as necessary. In a similar way, real values are transformed to binary using thermometer encoding, as this seems most likely to preserve information useful for discrimination. Let T be an integer greater than 1. Thermometer encoding of real values on T bits consists of normalizing all values to the interval [0,1] and converting each normalized value x to a bit-string of xT (rounded down) 1s followed by trailing 0s as needed.

The following briefly describes each dataset.

- **Mirror Symmetry**. In this problem, the system must separate patterns that are symmetrical about their center from those that are not. Here, patterns have 30 inputs. The first 15 inputs of each pattern are set randomly with each input having a probability of .5 of being 0 or 1. Then, with probability .5 again, each pattern is assigned to class 0 or 1. Patterns in class 1 are completed by setting the remaining 15 inputs symmetrically to the first 15 inputs about the pattern's center, while patterns of class 0 are completed by randomly setting the remaining 15 inputs with each input having a probability of .5 of being 0 or 1, ensuring that the pattern is not symmetrical about its center. The results reported here are based on 100 training patterns and 400 test patterns.

- **Shift Detection**. In this problem, the system must separate patterns whose right-hand side is a circular left shift of their left-hand side from those whose right-hand side is a circular right shift of their left-hand side. Here, patterns have 20 inputs. The first 10 inputs of each pattern are set randomly with each input having a probability of .5 of being 0 or 1. Then, with probability .5 again, each pattern is assigned to class 0 or 1. Patterns in class 1 are completed by setting the remaining 10 inputs identically to the first 10 but circularly shifted one position to the left, while patterns of class 0 are completed by setting the remaining 10 inputs identically to the first 10 inputs but circularly shifted one position to the right. The results reported here are based on 100 training patterns and 1,000 test patterns.
- **MONK's Problems**. The three MONK's problems have no real meaning, other than being explicitly relational tasks. All three problems have the same domain consisting of 6 input attributes, $a_1, \ldots, a_6$, all nominal. Attributes a_1, a_2 and a_4 range over $\{1, 2, 3\}$. Attributes a_3 and a_6 range over $\{1, 2\}$. Attribute a_5 ranges over $\{1, 2, 3, 4\}$. The target concepts, which vary in complexity, are as follows for each problem.
  - MONK1: $a_1 = a_2 \vee a_5 = 1$
  - MONK2: Exactly two of $\{a_1 = 1, a_2 = 1, a_3 = 1, a_4 = 1, a_5 = 1, a_6 = 1\}$
  - MONK3: $(a_5 = 3 \wedge a_4 = 1) \vee (a_5 \neq 4 \wedge a_2 \neq 3)$

  Each problem consists of 432 patterns. All of the patterns are in the test set, while the training sets consist of 124, 169 and 122 randomly selected patterns, respectively.
- **Zoo**. This is a small, simple animal classification task. There are 101 instances of animals, described by 16 inputs representing features such as whether the animal has feathers, lays eggs, or is venomous. 15 of the inputs are Boolean, 1 is integer valued and represents the number of legs (0, 2, 4, 5, 6, or 8). The integer-valued input is encoded using feature-based encoding. In its original form, the problem includes 7 output classes. We have transformed into a binary task by arbitrarily grouping classes 1 and 2 in one group, and classes 3 to 7 into another. The results reported are based on 10-fold cross-validation.
- **Glass**. In this problem, the system must identify glass types based on content. There are 9 inputs, all real-valued. All inputs have been thermometer-encoded with T=20. There are a total of 214 patterns and the results reported here are based on 10-fold cross-validation. The original database has 7 classes of glass. However, these classes are naturally combined into two general classes: window and non-window glass, making the problem a binary classification task that i-AA1* can handle directly.
- **Sonar**. This is the well-known rock vs mine discrimination problem [17]. Each pattern consists of 60 real-valued numbers in the range [0,1], representing the energy within a particular frequency band, integrated over a certain period of time. The values have been thermometer-encoded with T=10. There are a total of 208 patterns and the results reported here are based on 10-fold cross-validation.
- **Mushroom**. This is a large, but relatively simple task consisting in deciding whether a mushroom is edible or poisonous on the basis of 20 physical characteristics. All inputs are nominal, with some missing values in one of the inputs. Instead of the longer feature-based encoding, input values have simply been first transposed into integers and then translated into their binary equivalent values.[2] There are a total of 8,124 patterns and the results reported here are based on 10-fold cross-validation.

Table I reports predictive accuracy (PA) on the test set together with the size of the final network in number of nodes, the ratio of of the number of nodes to the number of training patterns. The numbers in parentheses are standard deviations. To account for the effects of order-dependency, all experiments (including each fold in the cross-validation settings) are repeated 10 times with a new random ordering of the training set.

TABLE I.
EXPERIMENTAL RESULTS

	PA	Size	Nodes/Patterns
Mirror Symmetry	75.6 (4.8)	90 (8)	0.90
Shift Detection	86.6 (4.0)	55 (10)	0.55
MONK1	92.7 (3.9)	56 (17)	0.45
MONK2	71.1 (2.6)	176 (15.5)	1.04
MONK3	91.9 (1.7)	48 (7.7)	0.39
Zoo	99.3 (1.6)	10 (5.3)	0.11
Glass	92.9 (3.0)	47 (9.2)	0.24
Sonar	71.6 (6.5)	140 (12.5)	0.75
Mushroom	100 (0.0)	54 (13.9)	0.007

The accuracy results for i-AA1* are comparable to those reported in the literature for other relevant learning algorithms, as illustrated in Table II.

VI. PRIOR KNOWLEDGE

As mentioned in section III-A, one of the features of i-AA1* is its ability to handle both examples and precepts, thus allowing prior knowledge to be combined with learning. Indeed, precepts capture explicit prior knowledge about the relevance of certain attributes' values to the prediction of the target attribute. That is, for a given value of the target attribute, a precept encodes which attributes are critical together with their associated values, and which attributes are irrelevant. Alternatively, from a logical standpoint, a precept encodes the minimum set of sufficient conditions (i.e., premise) for a particular conclusion to be derived. Within the context of a particular inductive task, precepts also serve as useful learning biases that speed up learning, as shown by the following theorem.

[2]We also performed minor data cleaning, discarding one input whose values are all the same and values of inputs that do not appear in the data set.

TABLE II.
COMPARATIVE RESULTS ON ACCURACY

		Accuracy
Mirror Symmetry		
	i-AA1*	75.6
	Target Switch [5]	80.4
Shift Detection		
	i-AA1*	86.6
	Backprop [32]	83.5
	Target Switch [5]	91.4
MONK1		
	i-AA1*	92.7
	ID3 [40]	98.6
	Backprop [40]	100
MONK2		
	i-AA1*	71.1
	ID3 [40]	67.9
	Backprop [40]	100
MONK3		
	i-AA1*	91.9
	ID3 [40]	94.4
	Backprop [40]	97.2
Glass		
	i-AA1*	92.9
	Target Switch	81.3
	C4.5	93.5
	Backprop	93.9
Sonar		
	i-AA1*	71.6
	Nearest-N [17]	82.7
	Target Switch [5]	85.0
	Backprop [17]	90.4
Mushroom		
	i-AA1*	100
	C4.5	100
	Backprop	100

Theorem 3: Let p be a pattern and P be a precept that subsumes p. Let $iNet$ be an i-AA1* network to which P has been presented previously. Then, the output of $iNet$ for p is correct.
The proof follows immediately from Theorem 2 and the fact that P subsumes p.

Two applications, from the UCI Repository [31], are used here to show the effect of precepts on learning time and network size.

- **Lenses**. In this simple application, the system must learn whether a patient should be fitted with hard contact lenses, soft contact lenses or no contact lenses. The problem is easily turned into a binary classification task by removing the distinction between hard and soft contact lenses. There are 4 nominal attributes and 24 patterns. Three rules can easily be identified from the patterns, which we use as precepts. Results are shown in Table III. Each row shows how accuracy increases as precepts are added to the 24 patterns during learning (under 10-fold cross-validation). No results are shown for time as the dataset is too small for these to be meaningful.

- **Shuttle Landing Control**. In this problem, the system must determine the conditions under which an autolanding would be preferable to manual control of a spacecraft. There are 6 nominal attributes, including wind direction and visibility. The raw dataset consists of 9 precepts and 6 specific patterns. When expanded, by replacing each precept with the patterns it represents, the dataset contains a total of 253 patterns. Results are shown in Table IV. The Base result corresponds to learning from the original dataset and testing on the expanded one. The other rows correspond to learning (under 10-fold cross-validation) from the expanded dataset in the presence of an increasing number of precepts. As expected, accuracy rises, training time is reduced and network size decreases as more precepts are provided.

TABLE III.
EXPERIMENTAL RESULTS WITH LENSES

	PA	Size
None	72.3 (15.8)	13 (1.8)
1	72.7 (11.7)	13 (2.0)
2	82.8 (9.9)	13 (1.9)
3	80.8 (9.9)	13 (1.9)

TABLE IV.
EXPERIMENTAL RESULTS WITH SHUTTLE LANDING CONTROL

	PA	Size	Time
Base	100.0 (0.0)	21 (1.7)	0.0
None	97.2 (2.6)	38 (9.0)	0.82
1	97.8 (2.1)	37 (9.1)	0.87
2	97.7 (2.4)	33 (7.6)	0.49
3	97.6 (1.7)	26 (5.7)	0.29
9	98.8 (0.9)	20 (3.7)	0.08

- **House Votes 1984**. In this problem, the system must determine whether a member of the U.S. House of Representatives is a Democrat or a Republican from his/her voting pattern across 16 key issues identified by the Congressional Quarterly Almanac for the 98th Congress. The dataset contains 435 records with many missing values, where missing here means that no vote was recorded. In addition with the raw data, the problem comes with detailed information about class predictiveness and predictability, which lends itself nicely to the derivation of simple precepts. Here, we use two precepts obtained as follows.
  - Given that $P(Democrat|physician-fee-freeze = nay) = 0.99$ and $P(physician-fee-freeze = nay|Democrat) = 0.92$, our first precept is "if physician-fee-freeze = nay then Democrat."
  - Given that $P(Democrat|adoption-of-the-budget-resolution = yea) = 0.91$ and $P(adoption-of-the-budget-resolution = yea|Democrat) = 0.87$, our second precept is "if adoption-of-the-budget = yea then Democrat."

  To satisfy i-AA1*'s consistency assumption, we remove the 35 records that would cause inconsistencies with our selected precepts. Results, based on 10-fold cross-validation, are shown in Table V. Although there is no significant change in accuracy, there is a

significant reduction in the size of the final network and in the time required to induce the network.

TABLE V.
EXPERIMENTAL RESULTS WITH HOUSE VOTES 84

	PA	Size	Time
None	97.1 (1.6)	38 (4.9)	10.24
2	97.0 (na)	34 (4.4)	7.68

Recall that in i-AA1*, precepts must be correct (i.e., consistent with the training data). This requirement is relaxed in other precept-guided learning systems such as FLARE [15]. Despite the gains in accuracy, training time and network size, there still remain some inefficiencies in i-AA1* since the node tables store the output of each node for all patterns, including those subsumed by the precepts.

Another, less explicit, way of using prior knowledge is in encoding heuristics for node addition. Instead of recruiting new variables based on their ability to discriminate, one could consider using prior information given by the user regarding the relative value or relevance of each variable, or use heuristics such as information gain.

VII. CONCLUSION

In this paper, we have detailed i-AA1*, a constructive, incremental learning algorithm for binary classification tasks. Since it implements symbolic knowledge (i.e., simple logical implications) on a neural network structure, i-AA1* can be viewed as a special instance of the class of hybrid symbolic connectionist learning systems. An excellent survey of such systems was recently published [28].

The basis of i-AA1* is the successive presentation of training patterns and their discrimination from all previously seen discordant patterns. Convergence is guaranteed and generalization is achieved as a result of a strong bias in favor of parsimonious networks.

Key features of i-AA1* include:

- Self-organization (i.e., adaptive network architecture),
- Incremental learning,
- Prior knowledge and
- Low-order polynomial complexity.

Empirical studies demonstrate the usefulness and validity of the algorithm on a variety of tasks. The explicit use of prior knowledge extends the system's applicability. As witnessed by current trends in research, adaptive hybrid connectionist and symbolic systems, as well as systems performing induction from examples and prior knowledge, seem to hold promise.

REFERENCES

[1] Y. Abu-Mostafa. Learning from Hints in Neural Networks. *Journal of Complexity*, **6** (1990) 192-198.

[2] Y. Abu-Mostafa. Hints. *Neural Computation*, **7** (1995) 639-671.

[3] O. Aran. Incremental Neural Network Construction Algorithms for Training Multilayer Perceptrons. MS Thesis, Bogazici University, Turkey, 2002.

[4] C. Barker and T.R. Martinez. Proof of correctness for ASOCS AA3 networks. *IEEE Transactions on Systems, Man, and Cybernetics*, **24**(3) (1994) 503-510.

[5] C. Campbell and C.P. Vicente. The target switch algorithm: A constructive learning procedure for feed-forward neural networks. *Neural Computation*, **7** (1995) 1245-1264.

[6] B. Copeland and D. Proudfoot. Alan Turing's forgotten ideas in computer science. *Scientific American*, April 1999, 77-81.

[7] J.L. Elman. Incremental Learning, or The Importance of Starting Small. Technical Report CRL 9101, University of California, San Diego, Center for Research in Language, (1991).

[8] S. Fahlman and C. Lebiere. The cascade correlation architecture. In *Advances in Neural Information Processing Systems 2*, (1990) 524-532.

[9] D. Fisher. Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning*, **2** (1987) 139-172.

[10] M. Frean. The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation*, **2** (1990) 198-209.

[11] S.I. Gallant. Three constructive algorithms for network learning. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, (1986) 652-660.

[12] C. Giraud-Carrier and T. Martinez. Using precepts to augment training set learning. In *Proceedings of the First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, (1993) 46-51.

[13] C. Giraud-Carrier and T. Martinez. Analysis of the convergence and generalization of AA1. *Journal of Parallel and Distributed Computing*, **26** (1995) 125-131.

[14] C. Giraud-Carrier and T. Martinez. AA1*: A dynamic incremental network that learns by discrimination. In *Proceedings of the Second International Conference on Artificial Neural Networks and Genetic Algorithms*, (1995) 41-45.

[15] C. Giraud-Carrier and T. Martinez. An Integrated Framework for Learning and Reasoning. *Journal of Artificial Intelligence Research*, **3** (1995) 147-185.

[16] C. Giraud-Carrier. A note on the utility of incremental learning. *AI Communications*, **13**(4) (2000) 215-223.

[17] R.P. Gorman and T.J. Sejnowski. Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets. *Neural Networks*, **1** (1988) 75-89.

[18] L.O. Hall and S.G. Romaniuk. A Hybrid Connectionist Symbolic Learning System. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, (1990) 783-788.

[19] Y. Jin and B. Senfhoff. Knowledge Incorporation into Neural Networks from Fuzzy Rules. *Neural Processing Letters*, **10** (1999) 231-242.

[20] S.Y. Kung and J.N. Hwang. An algebraic projection analysis for optimal hidden units size and learning rates in back-propagation learning. In *Proceedings of the IEEE International Conference on Neural Networks*, Vol. 1 (1988) 363-370.

[21] B. Lemarié and A-G. Debroise. A dynamical architecture for a radial-basis function network. In *Proceedings of the Second International Conference on Artificial Neural Networks and Genetic Algorithms*, (1995) 305-308.

[22] D. Martinez and D. Estève. The offset algorithm: Building and learning method for multilayer neural networks. *Europhysics Letters*, **18**(2) (1992) 95-100.

[23] T.R. Martinez. Adaptive self-organizing networks. Ph.D. Thesis, University of California, Los Angeles, (1986).

[24] T.R. Martinez and J.J. Vidal. Adaptive Parallel Logic Networks. *Journal of Parallel and Distributed Computing*, **5**(1) (1988) 26-58.

[25] T.R. Martinez. Consistency and generalization in incrementally trained connectionist networks. In *Proceedings of the International Symposium on Circuits and Systems*, (1990) 706-709.

[26] T.R. Martinez and D.M. Campbell. A Self-Adjusting Dynamic Logic Module. *Journal of Parallel and Distributed Computing*, **11**(4) (1991) 303-313.

[27] T.R. Martinez and D.M. Campbell. A Self-Organizing Binary Decision Tree for Incrementally Defined Rule Based Systems. *IEEE Transactions on System, Man, and Cybernetics*, **21**(5) (1991) 1231-1238.

[28] K. McGarry, S. Wermter and J. MacIntyre. Hybrid Neural Systems: From Simple Coupling to Fully Integrated Neural Networks. *Neural Computing Surveys*, **2** (1999) 62-93.

[29] G. Mitchandani and W. Cao. On hidden nodes for neural nets. *IEEE Transactions on Circuits and Systems*, **36**(5) (1989) 661-664.

[30] T.M. Nabhan and A.Y. Zomaya. Toward generating neural network structures for function approximation. *Neural Networks*, **7**(1) (1994) 89-99.

[31] D.J. Newman, S. Hettich, C.L. Blake and C.J. Merz. UCI Repository of Machine Learning Databases [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, (1998).

[32] S.J. Nowlan and G.E. Hinton. Simplifying Neural Networks by Soft Weight-sharing. *Neural Computation*, **4** (1992) 473-493.

[33] R.G. Parekh, J. Yang and V.G. Honavar. MUpstart - A Constructive Neural Network Learning Algorithm for Multi-Category Pattern Classification. In *Proceedings of the IEEE/INNS International Conference on Neural Networks*, (1997) 1924-1929.

[34] R.G. Parekh and V.G. Honavar. Constructive Theory Refinement in Knowledge Based Neural Networks. In *Proceedings of the International Joint Conference on Neural Networks*, (1998) 2318-2323.

[35] R. Polikar, L. Udpa, S. Udpa and V. Honavar, V. Learn++: An Incremental Learning Algorithm for Multi-Layer Perceptron Networks. *IEEE Transactions on Systems, Man, and Cybernetics*, **31**(4) (2001) 497-508.

[36] J. Siesma and R.J.F. Dow. Neural net pruning - why and how. In *Proceedings of the IEEE International Conference on Neural Networks*, Vol. 1, (1988) 325-333.

[37] S. Suddarth and A. Holden. Symbolic Neural Systems and the Use of Hints for Developing Complex Systems. *International Journal of Man-Machine Studies*, **35**(3) (1991) 291-311.

[38] R. Sun. A Connectionist Model for Commonsense Reasoning Incorporating Rules and Similarities. *Knowledge Acquisition*, **4** (1992) 293-321.

[39] R. Sun. An Efficient Feature-Based Connectionist Inheritance Scheme. *IEEE Transactions on Systems, Man, and Cybernetics*, **23**(2) (1993) 512-522.

[40] S.B. Thrun et al. The MONK's problems: A Performance Comparison of Different Learning Algorithms. Technical Report CS-CMU-91-197, Carnegie Melon University, (1991).

[41] G.G. Towell, J.W. Shavlik and M.O. Noordewier. Refinement of Approximate Domain Theories by Knowledge-Based Neural Networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, (1990) 861-866.

[42] G.G. Towell and J.W. Shavlik. Knowledge-Based Artificial Neural Networks. *Artificial Intelligence*, **70**(1-2) (1994) 119-165.

[43] P. Utgoff. ID5: An incremental ID3. In *Proceedings of the Fifth International Workshop on Machine Learning*, (1988), 107-120.

[44] Z. Wang, C. Di Massimo, M.T. Tham and A.J. Morris. A procedure for determining the topology of multilayer neural networks. *Neural Networks*, **7**(2) (1994) 291-300.

Christophe Giraud-Carrier is currently an Associate Professor in the Department of Computer Science at Brigham Young University (BYU). He received the BS, MS, and PhD degrees in Computer Science from BYU in 1991, 1993, and 1994, respectively.

He was Senior Lecturer in the Department of Computer Science at the University of Bristol, UK (1994-2001) and a Senior Manager at ELCA Informatique, Switzerland (2001-2004). He currently directs the Data Mining laboratory in the BYU PhD program, and has published widely in meta-learning, neural networks, and data mining.

Tony Martinez is a Professor and Chair of the Computer Science Department at Brigham Young University (BYU). He received the BS in Computer Science from Brigham Young University in 1982, and the MS and PhD in Computer Science from UCLA in 1983 and 1986, respectively.

He currently directs the neural network and machine learning laboratory in the BYU PhD program. He has published over one hundred refereed papers in the areas of machine learning, neural networks, and non von Neumann computing methods.