2006-07-26

# A Constructive Incremental Learning Algorithm for Binary Classification Tasks

Christophe G. Giraud-Carrier
cgc@cs.byu.edu

Tony R. Martinez
martinez@cs.byu.edu



**Original Publication Citation**
Giraud-Carrier, C., and Martinez, T. R., "A Constructive Incremental Learning Algorithm for Binary Classification Tasks", Proceedings of SMCals/6, pp. 213-218, 26.

# A Constructive Incremental Learning Algorithm for Binary Classification Tasks

Christophe Giraud-Carrier and Tony Martinez
Department of Computer Science
Brigham Young University
Provo, UT 84602
Email: {cgc,martinez}@cs.byu.edu

***Abstract*— This paper presents $i$-AA1$^\star$, a constructive, incremental learning algorithm for a special class of weightless, self-organizing networks. In $i$-AA1$^\star$, 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. Empirical generalization results on both toy problems and more realistic tasks demonstrate promise.**

## I. Introduction

Connectionist learning 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 sources 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 [1], [2], [3], [4].

Over the past decade however, many researchers have tried to overcome this difficult issue by focusing their efforts on the design of constructive learning methods (e.g., see [5], [6], [7], [8], [9], [10], [11]). In addition to modifying the connections' weights, constructive algorithms also evolve the network's topology. Hence, the topology is no longer an 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.

1-4244-0166-6/06/$20.00 (c)2006 IEEE

Adaptive Self-Organizing Concurrent Systems (ASOCS) define a special class of weightless, dynamic networks [13]. ASOCS networks learn by adapting their nodes' functions and by modifying their overall topology. ASOCS networks are parallel logic networks. 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 [14], AA2 [15] and AA3 [16]. Although all three algorithms have been shown to converge on any arbitrary Boolean pattern set, only AA1 exhibits meaningful out-of-sample generalization [17], [18].

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$^\star$, 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 [19]. Here, we make an explicit assumption of consistency, thus enabling true incremental learning [20], and demonstrate the algorithm's ability to exploit prior factual knowledge in learning. This new, incremental version of AA1$^\star$, called $i$-AA1$^\star$, requires less memory and is more computationally efficient than both of its predecessors.

The paper is organized as follows. Section II provides an overview of $i$-AA1$^\star$'s underlying architecture and learning paradigm. Section III reports experimental results on several synthetic datasets and more realistic applications. Section IV shows how prior knowledge is naturally incorporated in $i$-AA1$^\star$'s learning and presents experimental results that demonstrate the utility of prior knowledge in increasing parsimony and predictive accuracy. Finally, section V concludes the paper.

## II. $i$-AA1$^\star$

In this section, we first show how training patterns are represented and generalized. We then present $i$-AA1$^\star$'s architectural constraints and show how the network implements knowledge

in a distributed fashion across all the nodes. Finally, we detail the learning algorithm.

### A. Knowledge Representation

The target applications of $i$-AA1$^\star$ are binary classifications of Boolean-valued patterns. A training pattern consists of a set of values (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.

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

Fig. 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.

### B. Knowledge Implementation

Architecturally, the structure and connectivity of an $i$-AA1$^\star$ 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$^\star$ 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 | |

Fig. 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$^\star$ 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.

Two special kinds of nodes are defined here as they are central to $i$-AA1$^\star$'s learning. 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. 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.

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$^\star$'s learning algorithm.

### D. Learning Algorithm

As mentioned earlier, the target applications of $i$-AA1$^\star$ 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$^\star$ produces a sequence $iNet_1, iNet_2, \ldots, iNet_m$ of networks such that $iNet_{k+1}$ depends only on $iNet_k$ and $p_{k+1}$, for all $1 \leq k \leq m-1$. 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$^\star$ is shown in Figure 3. We assume that $m > 1$.

Step 1 is executed only once to initialize the network. So long as they remain unconnected, input nodes do not really

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

Fig. 3. $i$-AA1$^\star$ Learning Algorithm.

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 $\mid S \mid > 1$
   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 $NTN$.

Fig. 4. Node Combination

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$^\star$ 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.[1] 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. The procedure followed by $i$-AA1$^\star$ 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 network successively recruits the node that discriminates the incoming pattern from the largest number of remaining, non-discriminated, discordant patterns. This greedy selection process goes on until no further non-discriminated, discordant patterns can be accounted for.

Because the search through the network is essentially best-first, the set of selected nodes may not be smallest. Hence, the greedy selection process is repeated several times (here, 5), each time leaving out the node that first won the competition in the previous iteration. At the conclusion of this "optimization" loop, the smallest set of selected nodes with maximal discrimination is chosen. Empirical studies suggest that, in most cases, the smallest set is achieved after 2 or 3 iterations only. Because the number of 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.

Once a sufficient set of nodes has been obtained by selection, the nodes are combined in pairs as shown in Figure 4.

[1]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, this makes these nodes' discriminating capacities available for use in the constructive mechanisms of step 2b.

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. 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. 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 the correctness of the aforementioned procedure can only be guaranteed when there is no inconsistency (or noise) among the training patterns. At the cost of storing all training patterns and significant computational overhead, AA1 and AA1$^\star$ implement a simple, chronology-based method to handle inconsistencies online [18], [21]. By contrast, $i$-AA1$^\star$ assumes that the set of training patterns is consistent. Although somewhat restrictive, this assumption results in significant memory and computational savings.

## III. Experimental Results

This section reports some experimental results using $i$-AA1$^\star$ 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 [22]. The other tasks are drawn from the UCI repository [23].

Since $i$-AA1$^\star$ 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$^\star$ 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$^\star$ 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 and results 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 and results are based on 100 training patterns and 1000 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.
- **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$^\star$ can handle directly.
- **Sonar**. This is the well-known rock vs mine discrimination problem [24]. 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.

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 |
| **Glass** | 92.9 (3.0) | 47 (9.2) | 0.24 |
| **Sonar** | 71.6 (6.5) | 140 (12.5) | 0.75 |

The accuracy results are comparable to those widely reported in the literature for other learning algorithms. As expected, they tend to be lower for real-valued problems (i.e., glass and sonar) due to the binary encoding.

## IV. PRIOR KNOWLEDGE

An important aspect of $i$-AA1$^\star$ (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, $AC \rightarrow Z$ 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.[2] For historical reasons, we refer to training patterns with variables whose values are omitted as precepts [25], [26].

As per the above, 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.[3]

By using prior knowledge in the form of precepts together with raw patterns, $i$-AA1$^\star$ can effectively combine the intensional approach (based on features, expressed here by precepts) and the extensional approach (based on instances, expressed

[2] $i$-AA1$^\star$ 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.

[3] The philosophy and motivation underlying precepts are the same as that of (independently proposed) hints [34], [35]. 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$^\star$, 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 [36].

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$^\star$ network can readily be transformed back into its logical equivalent form and minimized using existing tools.

Two applications, from the UCI Repository [23], 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 II. 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.

TABLE II
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) |

- **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 III. 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.

Note that in $i$-AA1$^\star$, precepts must be correct (i.e., consistent with the training data). This requirement is relaxed in other precept-guided learning systems such as FLARE [26]. Despite the gains in accuracy, training time and network size, there still remain some inefficiencies in $i$-AA1$^\star$ since the node tables store the output of each node for all patterns, including those subsumed by the precepts.

TABLE III
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 |

## V. CONCLUSION

In this paper, we have detailed $i$-AA1$^\star$, 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$^\star$ 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 [27].

The basis of $i$-AA1$^\star$ 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$^\star$ 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] 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.

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

[3] 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.

[4] 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.

[5] C. Campbell and C.P. Vicente. The target switch algorithm: A constructive learning procedure for feed-forward neural networks. *Neural Computation*, **7** (1995) 1245-1564.

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

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

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

[9] 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.

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

[11] T.M. Nabhan and A.Y. Zomaya. Toward generating neural network structures for function approximation. *Neural Networks*, **7**(1) (1994) 89-99.
[12] B. Copeland and D. Proudfoot. Alan Turing's forgotten ideas in computer science. *Scientific American*, April 1999, 77-81.
[13] T.R. Martinez. Adaptive self-organizing networks. Ph.D. Thesis, University of California, Los Angeles, (1986).
[14] T.R. Martinez and J.J. Vidal. Adaptive Parallel Logic Networks. *Journal of Parallel and Distributed Computing*, **5**(1) (1988) 26-58.
[15] T.R. Martinez and D.M. Campbell. A Self-Adjusting Dynamic Logic Module. *Journal of Parallel and Distributed Computing*, **11**(4) (1991) 303-313.
[16] 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.
[17] 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.
[18] C. Giraud-Carrier and T. Martinez. Analysis of the convergence and generalization of AA1. *Journal of Parallel and Distributed Computing*, **26** (1995) 125-131.
[19] C. Giraud-Carrier and T. Martinez. AA1$^\star$: 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.
[20] C. Giraud-Carrier. A note on the utility of incremental learning. *AI Communications*, **13**(4) (2000) 215-223.
[21] T.R. Martinez. Consistency and generalization in incrementally trained connectionist networks. In *Proceedings of the International Symposium on Circuits and Systems*, (1990) 706-709.
[22] 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).
[23] 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).
[24] 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.
[25] 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.
[26] C. Giraud-Carrier and T. Martinez. An Integrated Framework for Learning and Reasoning. *Journal of Artificial Intelligence Research*, **3** (1995) 147-185.
[27] 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.
[28] 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.
[29] R. Sun. A Connectionist Model for Commonsense Reasoning Incorporating Rules and Similarities. *Knowledge Acquisition*, **4** (1992) 293-321.
[30] R. Sun. An Efficient Feature-Based Connectionist Inheritance Scheme. *IEEE Transactions on Systems, Man, and Cybernetics*, **23**(2) (1993) 512-522.
[31] 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.
[32] G.G. Towell and J.W. Shavlik. Knowledge-Based Artificial Neural Networks. *Artificial Intelligence*, **70**(1-2) (1994) 119-165.
[33] Y. Jin and B. Senfhoff. Knowledge Incorporation into Neural Networks from Fuzzy Rules. *Neural Processing Letters*, **10** (1999) 231-242.
[34] Y. Abu-Mostafa. Learning from Hints in Neural Networks. *Journal of Complexity*, **6** (1990) 192-198.
[35] Y. Abu-Mostafa. Hints. *Neural Computation*, **7** (1995) 639-671.
[36] 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.