1993-11-26

Towards a General Distributed Platform for Learning and Generalization

Brent W. Hughes
fourinchbunny@yahoo.com

Tony R. Martinez
martinez@cs.byu.edu

Original Publication Citation
Martinez, T. R., and Hughes, B., "Towards a General Distributed Platform for Learning and Generalization", Proceedings of the Conference on Artificial Neural Networks and Expert Systems ANNES'93, pp. 216-219, 1993.

BYU ScholarsArchive Citation
Hughes, Brent W. and Martinez, Tony R., "Towards a General Distributed Platform for Learning and Generalization" (1993). *All Faculty Publications*. 1176.
https://scholarsarchive.byu.edu/facpub/1176

Towards a General Distributed Platform for Learning and Generalization[†]

Tony R. Martinez
Brent W. Hughes

Computer Science Department, Brigham Young University, Provo, Utah 84602

Abstract

Different learning models employ different styles of generalization on novel inputs. This paper proposes the need for multiple styles of generalization to support a broad application base. The Priority ASOCS model (Priority Adaptive Self-Organizing Concurrent System) is overviewed and presented as a potential platform which can support multiple generalization styles. PASOCS is an adaptive network composed of many simple computing elements operating asynchronously and in parallel. The PASOCS can operate in either a data processing mode or a learning mode. During data processing mode, the system acts as a parallel hardware circuit. During learning mode, the PASOCS incorporates rules, with attached priorities, which represent the application being learned. Learning is accomplished in a distributed fashion in time logarithmic in the number of rules. The new model has significant learning time and space complexity improvements over previous models.

[†]This research supported in part by grants from Novell, Inc. and Word Perfect Corp.

1. Introduction

Generalization in a learning system is at best always a guess. The proper style of generalization is application dependent. Thus, one style of generalization may not be sufficient to allow a learning system to support a broad spectrum of applications [14]. Current connectionist models use one specific style of generalization which is implicit in the learning algorithm. We suggest that the type of generalization used be a self-organizing parameter of the learning system which can be discovered as learning takes place. This requires a) a model which allows flexible generalization styles, and b) mechanisms to guide the system into the best style of generalization for the problem being learned.

This paper overviews a learning model which seeks to efficiently support requirement a) above. The model is called Priority ASOCS (PASOCS) [9], which is a member of a class of models called ASOCS (Adaptive Self-Organizing Concurrent Systems) [5].

ASOCS models support efficient computation through self-organized learning and parallel execution. Learning is done through the incremental presentation of rules and/or examples. ASOCS models learn by modifying their topology. Data types include Boolean and multi-state variables; recent models support analog variables. The model incorporates examples or rules into an adaptive logic network in a parallel and self organizing fashion.

In processing mode, ASOCS supports fully parallel execution on actual inputs according to the learned rules. The adaptive logic network acts as a parallel hardware circuit during execution, mapping n input vectors into m output vectors, in a combinatoric fashion.

The overall philosophy of ASOCS follows the high level goals of current neural network models. However, the mechanisms of learning and execution vary significantly. The ASOCS logic network is topologically dynamic with the network growing to efficiently fit the specific application. Current ASOCS models are based on digital nodes. ASOCS also supports use of symbolic and heuristic learning mechanisms, thus combining the parallelism and distributed nature of connectionist computing with the potential power of AI symbolic learning. A proof of concept ASOCS chip has been developed [1].

ASOCS models have two significant advantages over other learning models:

a) they are guaranteed to learn any arbitrary set of legal rules;
b) learning time is both fast and bounded (linear with the depth of the network, and logarithmic with the number of nodes).

Application targets for ASOCS include adaptive logic devices, logical inference, robotics, embedded systems, and pattern recognition. A number of formal ASOCS models have been proposed to date [6,7,8].

Section 2 of this paper gives an example of how different generalization techniques can approach a problem. Section 3 presents an overview of PASOCS. Section 4 illustrates how flexible generalization can be supported. Section 5 concludes the paper.

2. Generalization

A critical goal in learning models is *generalization* -- the phenomenon that after learning, a system will with high probability respond correctly to input patterns it has not previously seen.

Two common approaches to generalization are *hamming* and *critical feature* generalization. Hamming, or similarity based generalization schemes, set the output of a novel instance according to how closely it matches a stored *prototype* or stable state in a system. This type of generalization is employed in such approaches as Hopfield networks [4], Boltzmann machines [3], competitive learning[12], and case-based reasoning [2]. Other approaches seek to discover general critical features which can direct generalization of novel inputs. These include

0-8186-4260-2/93 $03.00 © 1993 IEEE

Backpropagation [13], ID3 [11], the A^q algorithms [10], and ASOCS. One of the apparent powers of natural nervous systems is the ability to discriminate the current critical inputs from a massive barrage of total inputs. If you put your hand on a hot stove, then immediate retraction should take place regardless of the many other impinging input variables. Which of the styles is the best for actual applications? We propose that both (and others) are required if a broad range of applications are to be successfully learned.

As an example, assume {A, B, C} are Boolean variables and the following examples are given to a learning system.

A B C	=>Z
A B' C	=>Z
A B C'	=>Z
A B' C'	=>Z
A' B' C'	=>Z'

Let A'BC (A=0, B=1, C=1) be presented to the system after learning the five examples. Since there was no example given of A' B C, the system must do generalization and/or output a don't know. If a critical feature scheme is used, the system could use the correlation of A => Z and A' => Z' as a rule with A as a critical feature, since the value of Z is always the same as A. Thus, under the critical feature scheme the system would output Z'. If a hamming distance scheme is used, then A' B C differs by one variable from A B C => Z, but differs by at least two variables from all other examples. Thus, under a hamming distance scheme the system would output Z. Which method (if either of the two) is correct is application dependent.

3. Overview of Priority ASOCS

To give the reader the overall flavor of PASOCS processing, we informally describe a Prioritized ASOCS architecture. For a detailed presentation of the PASOCS model see [9]. For simplicity, PASOCS is here described with only Boolean variables. PASOCS naturally supports nominal variables and supports continuous variables through use of a preprocessing module. A Priority ASOCS is an ASOCS model which has the following attributes. PASOCS:

- processes inputs in the form of Boolean variables and outputs Boolean results;
- accepts rules made up of a conjunction of boolean inputs which imply a boolean output;
- learns new rules over time;
- automatically resolves rule conflicts;
- combines specific rules into more general rules where appropriate;
- maintains an associated priority with each rule.

We begin with the black box of figure 1, having four input variables {a,b,c,d} and the single output variable z. In general, an ASOCS maps *n* input variables into *m* output variables where the size of *m* and *n* do not need to be known a priori. The system allocates input and output lines as new variables are introduced in rules.

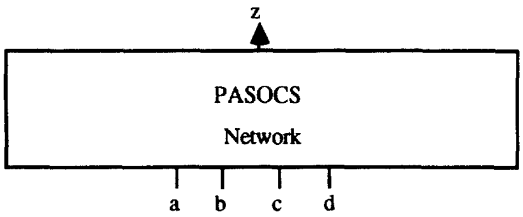

Figure 1 - PASOCS black box

Nodes and Rules: The black box of figure 2 contains nodes connected by a binary tree structure. Each node represents a rule. A rule is represented as a 3-tuple. The first element is a conjunction of boolean variables (the left hand side of the rule), the second element is the rule output (1 or -1), and the third element is the priority of the rule. The priority is a positive integer value with 1 being the lowest priority. The rule of node 1 in figure 2 has priority 2. It says to output (z=-1) whenever (a=1) and (b=1) and (c=0). Node 4 is currently empty.

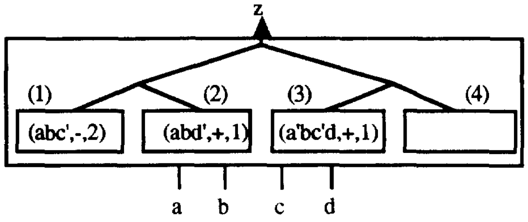

Figure 2 - Internal node representations

Execution: When the black box of figure 3 receives the input (a=1, b=1, c=0, d=0), it broadcasts these values down the binary tree structure to each node. Each node independently and in parallel compares itself to the input. If the input matches the rule, then the node passes up the binary tree a message consisting of the rule output and the rule priority. If the node does not match, no message is sent. *When two messages arrive at a junction in the tree, the one with higher priority (larger integer) is passed on while the other is discarded.* In the case of figure 3 and the given input, nodes 1 and 2 both fire. Since node 1 has priority 2, and node 2 has priority 1, the system outputs the result of node 1, which specifies an output of (z=-1).

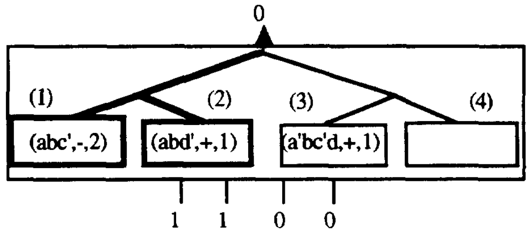

Figure 3 - PASOCS during input processing

Learning: When a new rule is presented to the system, it is broadcast down the tree to the nodes. Each node compares itself with the rule, and depending on how it matches, the node initiates an appropriate operation. The matching and operations occur independently and in parallel. These operations make up the learning algorithm. We present some examples to give a flavor for the basic learning algorithm. In the following, assume we wish to add the rule bd' => z, to the black box of figure 3. The following actions would take place. Though we present them sequentially, they would take place in parallel in an actual system.

Node Addition and Consistency Resolution: To resolve inconsistencies the system need only give the new rule a priority one greater than any rule it contradicts. The rule bd' => z is broadcast and all nodes which contradict the new rule (i.e. nodes which could be simultaneously matched for some input, but would cause the opposite output) send their priority up the tree. The highest priority arrives at the top and the new rule is added to the network with priority one higher than the highest inconsistent priority. In the case of figure 3, only node 1 (abc' = >z') is inconsistent with bd' = >z and it has priority 2. The system, therefore, adds bd' => z with priority 3, as shown in figure 4.

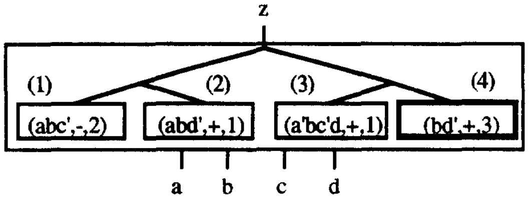

Figure 4 - Adding a new rule

Generalization: The new rule bd' => z has made node 2 redundant, since node 2 is matched in all cases where node 4 is matched. As part of the learning algorithm, the redundant node 2 deletes itself, as shown in figure 5.

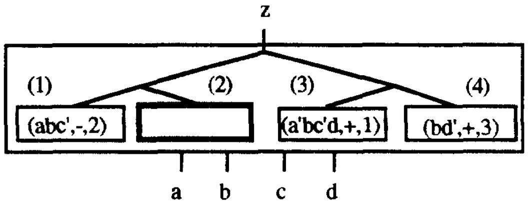

Figure 5 - After self deletion

The new rule bd' => z makes it possible for node 3 of figure 5 to generalize itself by removing the variable d. As part of the learning algorithm, node 3 minimizes itself, as shown in figure 6.

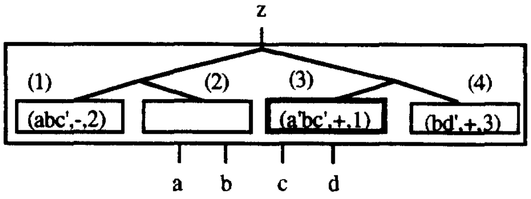

Figure 6 - After further generalization

This example gives a quick view of the style of learning mechanisms available in a PASOCS system. We now conclude the list of PASOCS attributes begun earlier:

- In Prioritized ASOCS (PASOCS), whenever a new rule is learned, at most one node need be added to the system.
- PASOCS is guaranteed to learn any set of valid rules.
- PASOCS need only see a rule once when learning a rule set.
- PASOCS does not unlearn old rules when learning new rules, except in the case of conflict resolution.
- PASOCS can support multiple output variables.

Due to the localist node-per-rule nature of PASOCS, each node is independent of every other node. Thus, any network topology can be chosen to best fulfill such needs as cost, speed, and fault tolerance. The particular topological instantiation proposed above is that of a binary tree. Due to this decision, we can state that the PASOCS:

- processes inputs in *O(log(n))* time, where *n* is the number of nodes (rules) in the network;
- learns individual rules in *O(log(n))* time, where *n* is the number of nodes (rules) in the network.

4. Flexible Generalization

The PASOCS model presented above generalizes in the sense of simplifying rules. However, no mechanism is defined for what should be output when the input environment matches no learned rule. Other ASOCS models [6] use a distributed mechanism to accomplish automatic generalization for novel inputs.

In PASOCS, the style of generalization can be both arbitrary and potentially dynamic. Current research is testing different styles of generalization in order to understand:

- Which generalization schemes support which styles of application.
- Which schemes can be efficiently implemented on PASOCS and similar platforms.
- Should generalization be accomplished by using multiple styles simultaneously. The result would be a parameterized combination of generalization techniques. Parameters

(generalization weights) are learned during training.

Following is a brief list of generalization techniques currently being studied with the PASOCS model. Distance metrics include not only variable differences but are a function of additional discovered feature information such as probability, typicality, etc.

- Hamming distance (number of variable mismatches).
- Percentage mismatch.
- Hamming with weighted variables (analog weights).
- Sums of nearest *M* prototypes for each output class.
- Weighted sets of discovered critical features (higher order variable groupings).
- Confidence level (CL) at nodes: this can be as simple as an analog weight which changes in a non-linear fashion as a function of inputs which contradict or reinforce the feature represented by the node. The CL is not only used as part of the generalization function, but also as part of the overall output representing the confidence of the network in the current generalization. Two important side effects of using the CL in ASOCS are:
  - a) Ability to handle noisy training sets.
  - b) Ability to do standard training set learning along with the normal ASOCS incremental learning.
- Direct instance and prototype generalization using both novel and AI techniques of structural prototype generalization.

Initial studies support the conclusion that multiple generalization styles aid in improving generalization accuracy. Also, the majority of the techniques mentioned above can be efficiently supported in the PASOCS model.

5. Conclusion

This paper has proposed the need for multiple styles of generalization to support a broad application base. The Priority ASOCS model was presented as a potential platform which can support multiple generalization styles. The significant features of PASOCS include a) guaranteed learning of any set of rules and b) fast execution and learning times (*O(log(n))*, where *n* is the number of rules. The advantages of PASOCS over previous ASOCS models are 1) at most one node is added for each instance learned and 2) the execution and learning time bounds are based on the number of rules, rather than the number of nodes.

Current research in this area includes:

- Empirical and analytical studies of how multiple generalization schemes can support applications.
- Schemes which guide the automatic discovery of the best combination of generalization mechanisms for the application during learning.

6. Bibliography

1. Chang, J. and J. J. Vidal, Inferencing in Hardware, *Proceedings of the MCC-University Research Symposium*, Austin, TX, (July 1987).
2. Hammond, K., CHEF, In *Inside Case-Based Reasoning*, Eds C.K. Riesbeck & R.C. Shank, Hillsdale, NJ, Erlbaum, 1989.
3. Hinton, G. E., Sejnowski, T.J., and Ackley, D.H., Boltzmann Machines: Constraint Satisfaction Networks that Learn, Tech Rep. No. CMU-CS-84-119, Carnegie-Mellon Univ.
4. Hopfield, J. J., Neural Networks and Physical Systems with Emergent Collective Computational Abilities, *Proc. Natl. Acad. Sci. USA*, vol 79, 2554-2558, April 1982.
5. Martinez, T. R., Adaptive Self-Organizing Concurrent Systems, *Progress in Neural Networks*, vol. 1, Ch. 5, 105-126, Ablex Publishing, 1990.
6. Martinez, T. R. and Vidal, J.J., Adaptive Parallel Logic Networks, *Journal of Parallel and Distributed Computing*, vol. 5, no. 1, 26-58, 1988.
7. Martinez, T. R. and Campbell, D. M., A Self-Adjusting Dynamic Logic Module, *Journal of Parallel and Distributed Computing*, vol. **11**, no. 4, 303-313, 1991.
8. Martinez, T. R. and Campbell, D. M., A Self-Organizing Binary Decision Tree for Incrementally Defined Rule Based Systems, *IEEE Transactions on System, Man, and Cybernetics*, vol. **21**, no. 5, 1231-1238, 1991.
9. Martinez, T. R., Hughes, B., and Campbell, D. M., Priority ASOCS, To appear in *Journal of Artificial Neural Networks*, 1993.
10. Michalski, R.S., A Theory and Methodology of Inductive Learning, *Artificial Intelligence*, 20, pp. 111-116, 1983.
11. Quinlan. J. R., Induction of Decision Trees, *Machine Learning*, 1, 81-106, 1986.
12. Rumelhart, D. and McClelland J., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* 1, MIT Press, Cambridge, MA 1986, Ch. 5.
13. Rumelhart, D. and McClelland J., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* 1, MIT Press, Cambridge, MA 1986, Ch. 8.
14. Wilson, D. R, and Martinez, T. R., The Importance of Using Multiple Styles of Generalization on Training Set Data, submitted.