Presented at *2nd IEEE International Conference on Neural Networks*, 1988.

ASOCS: A Multilayered Connectionist Network with Guaranteed Learning of Arbitrary Mappings

Tony R. Martinez
Computer Science Dept., 230 TMCB
Brigham Young University, Provo, Utah 84602

Abstract

This paper reviews features of a new class of multilayer connectionist architectures known as ASOCS (Adaptive Self-Organizing Concurrent Systems). ASOCS is similar to most *decision-making* neural network models in that it attempts to learn an adaptive set of arbitrary vector mappings. However, it differs dramatically in its mechanisms. ASOCS is based on networks of adaptive digital elements which self-modify using local information. Function specification is entered incrementally by use of rules, rather than complete input-output vectors, such that a processing network is able to extract critical features from a large environment and give output in a parallel fashion. Learning also uses parallelism and self-organization such that a new rule is completely learned in time linear with the depth of the network. The model guarantees learning of any arbitrary mapping of boolean input-output vectors. The model is also stable in that learning does not erase any previously learned mappings except those explicitly contradicted.

Introduction

Current technology trends, demanding applications, and limitations of sequential computation, have combined to rejuvenate research into connectionist and neurally inspired computational mechanisms. A new class of multilayer connectionist systems, ASOCS (adaptive self-organizing concurrent systems), was created with the goal of fulfilling the desired functionality of connectionist methods, while overcoming some of the drawbacks of current models. A number of specific ASOCS architectures have been proposed [6] and VLSI implementation and testing are underway [2].

An ASOCS system solves any arbitrary dynamic mapping of boolean input vectors to boolean output vectors. The atomic knowledge input to the system is known as an *instance* and is made up of a variable length boolean input vector and an associated target output. The instance input vector typically contains only the critical features, (a small subset of the total environment input), sufficient to decide the current output. Thus, a single instance represents many possible states of the environment. Any propositional boolean function can be represented by a set of instances. Instances are input to and learned incrementally by the system. A new instance may contradict earlier instances, in which case only the contradicted portions of older instances optionally become invalid. A new instance is broadcast to a multilayer network of adaptive digital nodes. Using only local information (the internal state of the system or nodes is invisible to both the outside and inside), nodes adjust their functions and interconnectivity in a parallel and self-organizing fashion in order to maintain a network which consistently fulfills all instances received by the system. The network can build itself from scratch with new interconnections being dynamically made between nodes, together with the ability for nodes no longer aiding in correct network function to *self-delete.* During processing, the network acts as a parallel hardware circuit mapping input states to output states. Instances can be continually input over the life of the system.

Both connectionist systems and applications differ in their needs and goals. Effective high level processing may require an integration of separate models, each processing in its domain of strength, in order to acquire an emergent multi-purpose mechanism. Although a simplistic and incomplete dichotomy, I classify current artificial neural systems into *categorization* systems and *decision* systems. Examples of categorization systems include adaptive resonance theory [1] and self-organizing feature maps [4]. These systems do unsupervised learning of categories and classifications of arbitrary inputs. They seek solutions in the important domain of building up reasonable categories from an unknown input environment, mapping similar things into similar classes. Another application is that of taking mapped inputs, such as those from a categorization system, and further mapping them into outputs (motor actions) or inferences. In this case, many inputs which are very dissimilar may need to be mapped into the same output class, while similar inputs could be scattered into various output classes. This is the realm of decision systems. These systems are required to do more arbitrary mappings and require some type of supervised learning (teacher, environment reward, etc.) Attempts at this type of system include backpropagation techniques [9] and Boltzmann machines [3]. There is certainly overlap between categorization and decision systems, and classifying a model as one or the other is a matter of degree. ASOCS falls into the realm of decision making systems, and thus can be most accurately compared and measured against that class of models. Perhaps more general systems will include a categorization portion which does unsupervised classification of inputs into usable classes, and which feeds its outputs to a decision making system which controls motor or other output units (figure 1).

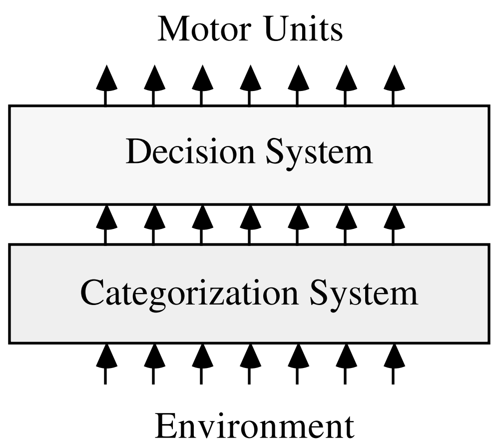

Figure 1 - Integrated categorization and decision systems

The goal of this paper is to summarize the features of the ASOCS system and view them in light of other decision type neural networks. The next section of this paper overviews these features and contrasts. The following section gives a summary of the ASOCS mechanisms of computation. The scope of this paper does not allow a detailed discussion of general mechanisms or of any single ASOCS model and detailed reports are found elsewhere [5,6,7,8].

Features of the Model

Although the goal functionality in ASOCS is similar to that of most decision making connectionist models, the method of operation and the features of the system are quite different. Much of this difference follows from the use of different types of atomic nodes. Most neural networks use a node which sums the products of inputs and weights, and then does some type of nonlinear function of that sum to attain an activation. Learning is accomplished through small changes in the weights. The ASOCS models uses as an atomic node, a two input boolean gate which can be set to any of the 16 possible functions of two inputs. Rather than gradient seeking algorithms of small weight changes, ASOCS must seek methods of learning through changing the function and interconnection between nodes. Another cause of disparity is the use of rule-like instances which require only the critical variables of any pattern to be shown, rather than the complete input/output vectors of a typical training set. In the following we discuss the domain of boolean input and output vectors.

Although multilayer nonlinear networks can theoretically implement arbitrary mappings of input/output vectors, learning algorithms have not yet been established which can guarantee that a desired mapping will actually be learned. Since current attempts at multilayer learning use variants of gradient descent or stochastic mechanisms, local minima and finite time constraints keep these models from always attaining a desired mapping. The learning algorithms of ASOCS models, on the other hand, have been shown to produce guaranteed mappings of any boolean function with arbitrarily many inputs and outputs. It must be noted that layered models using exponential numbers of nodes, such as a RAM, can be trained in a guaranteed fashion to learn arbitrary mappings. In fact, certain mappings would appear to *require* an exponential amount of either nodes, links, or weight resolution. These type of problems, however, are not in the domain of applications for which neural networks are intended. Most current models are based on an a priori static topology, and even when a problem domain contains a portion of highly complex mappings, (which *is* a common situation), the model requires the full exponential size demands. ASOCS, on the other hand, through use of a reconfigurable topology, can account in a dynamically customized manner for complex mappings, while maintaining a parsimonious representation for the entire system.

A second feature of the ASOCS system is that the time necessary for the network to accept a new instance and reconfigure to a consistent network is both bounded and scales well as the network grows. Learning time is $O(d)$ where d is the depth of the network and $O(log(n))$ where n is the number of nodes in the network. This compares very favorably with gradient descent techniques which require large numbers of iterations for the weights to arrive at a proper value.

Along with a fast bounded learning time, ASOCS has the additional feature of *stability*. Learning of a new instance does not alter the systems ability to continue properly fulfilling previously learned instances. (At the same time, portions of previous instances explicitly contradicted by a new instance can be modified). This is the opposite of *unlearning*, where adjusting weights to correctly fulfill the current pattern can undo and invalidate earlier adjustments made to previous pattern values. This leads to oscillation and/or many repetitions of the same input/output pairs in order to converge to a proper network. ASOCS is able to maintain stability without sacrificing a distributed mechanism of processing and representation.

An important goal of most neural network models is that of *generalization*. This is the phenomenon that after learning, a network will with high probability respond correctly to input patterns which it has not previously seen. In weight summing algorithms, similar input vectors typically generate similar outputs. The generalization is then in terms of the similarity of inputs; (an example measure is that of hamming distance). However, this is but one type of generalization. Correct generalization is application specific. In the ASOCS scheme, generalization is attained through the incomplete nature of instances, which allow specification of only critical variables for a given pattern, rather than an entire input vector. 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 one puts his hand to a hot stove, immediate retraction should take place regardless of the many other impinging input variables. Instances allow a natural mechanism to specify critical variables for specific situations, and generalization takes place in that many possible input vectors impinging during processing can match the single rule which has specified the critical variables. Thus a processing ASOCS system is able to extract the critical variables from a large input environment and make correct judgements although it has never before experienced the total current input environment before.

On the down side, the current author does not believe that the overall ASOCS system is very plausible as a biological model. The ability to extract critical inputs from an extended environment *is* considered as a notion which demands more attention in neural modeling. Also, the basic ASOCS tenant of simple broadcast to many unobservable nodes, with concomitant local communication and modification between those nodes, is seen as a strong analog of the endocrine/nervous system interaction. Finally, the ability to create new connections and nodes, together with the self-deletion of obsolete nodes, seems closer to the reality of developmental neurophysiology than initially fixed topological structures. However, an immediate analogy is not seen for the purely boolean nature of the data and functions of the atomic ASOCS node.

Summary of ASOCS Computational Mechanisms

In this section, a brief overview of ASOCS mechanisms and architecture are given. For detailed and formal presentation see [5,7]. Instances are first reviewed, followed by a description of the generic architecture and the primitive mechanisms necessary for ASOCS learning and processing.

Before discussing particular mechanisms, an attempt is here made to give a feel for the overall adaptive process in an ASOCS system. Since a number of formal systems have been proposed, there is no single overall learning process. At any time while a system is processing, a new instance can be entered into the system. The instance is then broadcast to the network by one of a few defined mechanisms. The i*nstance set,* the total function described by all learned instances, need not be explicitly stored, but can be implicitly stored by the functions and interconnections within the network. If the network does not fulfill the newly introduced instance, nodes within the network self-select to be part of the modification process whereby the network reconfigures to consistently fulfill the newly changed instance set. Selected nodes then independently and concurrently combine to form the new network. At this point, nodes no longer aiding in the correct functioning of the network delete themselves after sending messages to neighbor nodes for them to also delete themselves. This recursive self-pruning allows the network to maintain a parsimonious representation of its current function.

An instance is an if-then type rule where the antecedent is a conjunction of boolean variables and the consequent is a single boolean variable. Examples of instances are shown below.

$$A\ \overline{B} \Rightarrow Z$$

$$\overline{A}\ C\ D \Rightarrow \overline{Z}$$

$$Z\ \overline{D} \Rightarrow Y$$

In a given instance only those variables critical for deciding the output are necessary. For example, the first instance above states the Z will become high when A is high and B is low regardless of all other environment variables. It says nothing, however, about the case when A is low or B is high. Thus, it is somewhat different then typical propositional logic. Any arbitrary logic function can be represented by a set of instances. Note that output variables can be used in the antecedents of other instances, thus allowing a natural mechanism for both feedback and rule chaining.

An instance set is maintained *consistent.* By this it is meant that no two instances simultaneously matched by the environment can imply the opposite output value for a given output variable. When a new instance contradicts a previous instance, that portion of the previous instance which is contradicted is removed. For example, given the old instance $A \Rightarrow Z$ and the new instance $B \Rightarrow \overline{Z}$ the old instance would be modified to $A\ \overline{B} \Rightarrow Z$. This paradigm of precedence to the most recent instance need not be strictly adhered to and can be adjusted to the needs of a given application. The instance set may also be maintained *minimal*. This only affects parsimony of representation and is not essential for correct functioning.

The relation between the number of input variables and output variables is arbitrary. Variables do not need to be predefined in a system. Rather, not until the first time an input or output variable is seen in an instance is a data line dynamically bound to the variable. Thus, new variables can be introduced at any point in the life of the model. Figure 2 shows a black box representation of the basic system.

Figure 2 - Black Box ASOCS Structure

The overall block model representation of the standard ASOCS architecture is displayed in figure 3a. The multilayer network is represented by the logic network box. Input and output binders allow flexible binding of input and output variables to the network. The feedback path allows bound output variables to be fed back as inputs to the network. During the processing phase, these are the only active components and environmental data flows through the logic network in a parallel fashion to form outputs. During the adaptation phase the other parts of the system are active. (These parts may be concurrently active in self-testing, minimization, and other functions, but these are beyond the scope of this presentation). The adaption unit receives instances from the environment. It in turn causes broadcast of the new instance to the logic network. The adaption unit can broadcast information to all nodes in the network, but it has no ability to address any given node, or to observe the internal state of the logic network. Through the test path, the adaption unit can observe the outputs of the network, and the presentation path allows an optional mechanism for setting the inputs to the network during adaptation.

A single network node, many of which make up the logic network, is illustrated in figure 3b. A single node of the network, like the architecture in whole, is divided into two main units. The dynamic programmable logic module (DPLM) is a 2 input boolean

Figure 3a - ASOCS Architecture 3b) Single Adaptive Element

gate which can be set to any one of the 16 boolean functions of 2 inputs. Inputs to the DPLM come from children nodes lower in the network, while its output can proceed to any number of parent nodes. The control unit is active during adaptation and has the ability to change the function of its corresponding DPLM. The control unit also has the ability to read the broadcast mechanism, and send messages on the bidirectional control paths between its closest neighbors.

To this point, learning algorithms come from two broad classes. The first [7] requires some memory in the control unit of the individual nodes, storing the manner in which the node reacts to learned instances. During adaptation, nodes compare this ability with the constraints of the new instance and nodes with capability to discriminate the new instance from sets of old instances self-select, and afterward combine with other selected nodes to create a consistent network. Inspection of the control unit memory is then sufficient to ascertain the utility of the node and whether it should self-delete. These mechanisms have the disadvantage of memory growth proportional to the number of instances. They also maintain the consistent instance set in the adaption unit. The typical number of nodes in these networks is approximately equal to the number of instances.

The second class of algorithms [5,6] require only that the control unit store a static list of the variables upon which its DPLM currently functions. (Even this small portion of memory can be obviated by alternate broadcast techniques [2]). Each node can then do a direct comparison of its variable list with that of the broadcast new instance, and immediately fulfill a deterministic modification from a small set of rules depending on how it matches with the new instance. These algorithms require only negligible memory in the control unit independent of the number of instances. They also do not require explicit storage or maintenance of the instance set. An instance is simply broadcast to the network and the instance set is implicitly stored and maintained consistent in the adaptive network. The typical number of nodes in these networks is approximately two times the number of instances.

Summary

This paper has reviewed features of a new class of multilayered connectionist models called adaptive self-organizing concurrent systems (ASOCS). Features include guaranteed mapping of arbitrary boolean functions, bounded linear learning time, stability of previously learned patterns, and ability to extract critical features from a large environmental input. Initial fabrication and testing of VLSI chips is underway. Current research includes implementation mechanisms, expansion of knowledge input mechanisms to include analog inputs and rules with variable bindings, coupling with memory mechanisms, and use of feedback and sequencing.

Bibliography

1. Carpenter, G.A., and Grossberg, S., A massively parallel architecture fo a self-organizing neural pattern recognition machine, *Computer Vision, Graphics, and Image Processing,* **37,** pp. 54-115, (1987).
2. Chang, J. and J. J. Vidal, Inferencing in Hardware, *Proceedings of the MCC-University Research Symposium*, Austin, TX, (July 1987).
3. Hinton, G., Sejnowski, T, and D. Ackley, *Boltzmann Machines: Constraint Satisfaction Networks that Learn,"*Tech. Rep CMU-CS-84-119, CMU, Pittsburgh, PA, (1984).
4. Kohonen, T., *Self-organization and associative memory*, Springer Verlag, New York, (1984).
5. Martinez, T. R., *Adaptive Self-Organizing Logic Networks*, Ph.D. Dissertation, Technical Report - CSD 860093, University of California, Los Angeles, CA (May 1986).
6. Martinez T. R., Models of Parallel Adaptive Logic, *Proceedings of the 1987 IEEE Systems Man and Cybernetics Conference*, pp. 290-296, (October, 1987).
7. Martinez, T. R. and J. J. Vidal, Adaptive Parallel Logic Networks, *Journal of Parallel and Distributed Computing*, Vol. **5**, pp. 26-58, (1988).
8. Martinez, T.R., Digital Neural Networks, *Proceedings of the 1988 IEEE Systems Man and Cybernetics Conference*, (August, 1988).
9. Rumelhart, D. and McClelland, J., *Parallel Distributed Processing:*, Vol. I, pp. 318-362, MIT Press, (1986).