expert systems

IASTED

Proceedings of the Fifth IASTED International Symposium

EXPERT SYSTEMS AND NEURAL NETWORKS

Theory & Applications

Honolulu, Hawaii
August 16-18, 1989

EDITOR: M.H. Hamza

A Publication of
The International Association of Science and Technology for Development - IASTED

ISBN 0-88986-129-3

ACTA PRESS

ANAHEIM * CALGARY * ZURICH

TABLE OF CONTENTS

Fast Neural Learning Algorithms

J. Barhen **S. Gulati** **N. Toomarian**

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

ABSTRACT

A new theoretical framework for neural learning of nonlinear mappings is presented. The proposed methodology exploits a new class of mathematical constructs named terminal attractors which provide unique information processing capabilities to artificial neural systems. Terminal attractor representations are used not only to ensure infinite local stability of the encoded information, but also to provide a qualitative as well as quantitative change in the nature of the learning process. Typical performance improvements are in excess of three orders of magnitude over current state-of-the-art backpropagation techniques. In a significant departure from prior neuromorphic formulations our algorithms provide a framework for incorporating event-driven constraints in real-time, avoiding the necessity to retrain the network.

1. Introduction

A considerable effort has been devoted to the development of efficient computational methodologies for learning. Artificial neural networks, characterized as massively parallel, adaptive dynamical systems, provide an ideal framework for interacting with objects of the real world and its statistical characteristics in the same manner as biological systems do. Their paradigmatic strength for potential applications arises from their spontaneous emergent ability to achieve *functional synthesis*, and thereby learn *nonlinear mappings*, and abstract spatial, functional or temporal invariances of these mappings [1,3,5,8,9,10,11]. Thus, relationships between multiple continuous-valued inputs and outputs can be established, based on a presentation of a large number of *a priori* generated representative examples. Once the underlying invariances have been learned and encoded in the topology and strengths of the synaptic interconnections, the neural network can generalize to solve arbitrary problem instances. Although in the past, a number of neural algorithms have been proposed for functional approximation, attention has largely focussed on the back-propagation algorithm because of its simplicity, generality and the promise that it has shown in regard to various applications. However, the increasing perception that back-propagation is too slow to be relevant to most real-world problems has led to the development of a number of variant algorithms. Lapedes and Farber [8] proposed a master-slave network with sigmoidal nonlinearities to approximate a continuous time series for forecasting. Pineda [11] extended the methodology by deriving a recurrent generalization to back-propagation networks operating in continuous time. In a similar vein, Pearlmutter [10] constructed a procedure for approximating trajectories by minimizing an error functional between output and targeted temporal trajectories. More recently, Williams and Zipser [12] proposed a real-time learning algorithm for training recurrent, continually updated networks to handle temporal tasks.

In a radically different approach, we propose to use a new mathematical construct, i.e., terminal attractor dynamics [1,2,3,13,14] to acquire nonlinear mappings. Terminal attractor representations are used not only to ensure infinite local stability of the encoded information, but also to provide a qualitative as well as quantitative change in the nature of the learning process. In particular, the resulting loss of Lipschitz conditions at energy function minima yields a dramatic increase in speed of learning. A fundamental problem in neural learning methodologies based on dynamical systems concerns the stability of the activation network as synaptic weights evolve during training. Previous approaches [1,8,11] do not guarantee stability. Here, we introduce the concept of "virtual" terminal attractors which yields an unconditionally stable neurodynamics.

2 Network Specification

Consider a densely connected neural network with N graded-response neurons operating in continuously sampled time. To acquire a nonlinear transformation, ζ, from a $\aleph_X$-dimensional input domain to the $\aleph_Y$-dimensional output space, the network is topographically partitioned into three mutually exclusive regions. The partition refers to a set of input neurons, S_X, that receive the input components, an output set S_Y, which provides the desired output components and a set of "hidden" neurons, S_H, that encode the representation of the ζ-mapping. The network is presented with K randomly sampled training vector-pairs of input- and output-space coordinates.

We formalize the neural network as an adaptive dynamical system whose temporal evolution is represented

by the following coupled differential equations

$$\dot{u}_n + \kappa\, u_n = \sum_m T_{nm}\, \varphi_\gamma(u_m) + {}^k I_n \qquad (2.1)$$

where u_n represents the *mean soma potential* of the nth neuron and T_{nm} denotes the synaptic coupling from the mth to the nth neuron. The constant κ characterizes the decay of neuron activity. The sigmoidal function $\varphi_\gamma(\cdot)$ modulates the neural response, with gain given by γ; typically, $\varphi_\gamma(z) = \tanh(\gamma \cdot z)$. Without loss of generality, γ can be set to unity in the sequel. The "source" term, ${}^k I_n$ encodes component-contribution by the attractors of the k-th training sample via the expression

$${}^k I_n = \begin{cases} [{}^k a_n - \varphi(u_n)]^\beta & \text{if } n \in S_X \cup S_Y \\ 0 & \text{if } n \in S_H \end{cases} \qquad (2.2)$$

The specific attractor coordinates are given by

$${}^k a_n \equiv {}^k x_n \quad \text{if } n \in S_X$$
$${}^k a_n \equiv {}^k y_n \quad \text{if } n \in S_Y$$

for $\{ {}^k\bar{x},\ {}^k\bar{y} \mid k = 1, \cdots, K\}$ taken from a training set scaled to the range $[-1, +1]$. In [1,3,13,14] we have shown that, for $\beta = (2i+1)^{-1}$ and i a strictly positive integer, such attractors have infinite local stability and provide opportunity for learning in real-time.

The neural network specified above constitutes a *dissipative nonlinear dynamical system*, the flow of which generally converges to a manifold of lower dimensionality in the phase space. It is well known [5,7,13] that such neuromorphic systems can store memory states or patterns in terms of fixed points of the network dynamics, such that initial configurations of neurons in some neighborhood or *basin of attraction* of that memory state will be attracted to it. Of crucial importance is to know how stable these attractors are, and, starting from an arbitrary network configuration, how fast they can be reached. In this vein, we exploit a novel concept in dynamical systems theory, i.e., the notion of *terminal attractor* , that subsequently will enable us to formalize an efficient neural learning algorithm. The concept of terminal attractors in neural networks was initially introduced by Zak [13] to obviate some of the above limitations in content-addressable memories, and further extended by Barhen et al. [1,2,3,6] and Zak [14] to computational learning. It has also been shown that incorporation of terminal attractors in the neural dynamics can be used for elimination of spurious states [13]. This property is critical in providing an accurate generalization ability during the operational phase of our proposed network. It will ensure that "interpolations" (or "extrapolations") over the input domain of ζ are not based on false attractors. In our neuromorphic framework, terminal attractor dynamics then provides a mechanism that can implicitly exploit the time-bounded terminality of phase trajectories and their locally infinite stability, thereby enabling an efficient and accurate solution to learning functional invariants.

3. Energy Function and Network Stability

Our basic operating assumption for the dynamical system defined by (2.1) is that at equilibrium, i.e., as $\dot{u}_n \to 0$, for $n = 1, \cdots, N$,

$$u_n \to {}^k\tilde{u}_n(T). \qquad (3.1)$$

The superscript $\sim$ will be used to denote quantities evaluated at steady state. This yields the fixed point equations

$$\kappa\, {}^k\tilde{u}_n = \sum_m T_{nm}\, \varphi({}^k\tilde{u}_m) + {}^k\tilde{I}_n \qquad (3.2)$$

Note that, in contradistinction to Hopfield [7], Pineda [11], and others [8,9,12] ${}^k I_n$, is a function of the state variable u_n and does not represent a constant external input bias to the network. It influences the system's degree of stability and provides a dynamically varying input modulation to the neuron, thereby enforcing convergence to fixed points in finite time, without affecting the location of existing static attractors. For an arbitrary synaptic matrix T, the asymptotic attractor contribution ${}^k\tilde{I}_n$ differs from zero. The key *objective of learning is then to adaptively evolve the interconnection topology of the neural network and determine the synaptic strengths,* so that the $S_X \longrightarrow S_Y$ mapping be accurately computed over the training set, in terms of the specified attractors; i.e., $\forall\, k = 1, \cdots, K,\ {}^k\tilde{I}_n = 0 \qquad \forall\ n \in S_X \cup S_Y$

To proceed formally with the development of a learning algorithm, we propose an approach based upon the minimization of a constrained "neuromorphic energy-like function" $E(T, \lambda)$ given by the following expression

$$E(T,\lambda) = \frac{1}{2} \sum_n \sum_m \omega_{nm}\, T_{nm}^2 + \frac{1}{\alpha} \sum_k \sum_n {}^k\lambda_n\, {}^k\Gamma_n^\alpha \qquad (3.3)$$

where

$${}^k\Gamma_n = \begin{cases} {}^k a_n - \varphi({}^k\tilde{u}_n) & \text{if } n \in S_X \cup S_Y \\ 0 & \text{if } n \in S_H \end{cases} \qquad (3.4)$$

Typically, positive values like $\frac{4}{3}$ and 2 are used for α. The weighting factor ω_{nm} is constructed in such a fashion as to favor locality of computation. The indices n, m span over all neurons in the network. Lagrange multipliers corresponding to the $k-n$th constraint are denoted by ${}^k\lambda_n$. The proposed objective function includes contributions from two sources. First, it enforces convergence of every neuron in S_X and S_Y to attractors corresponding to the components in the input-output training samples, thereby prompting the network to learn the underlying invariances. Secondly, it regulates the topology of the network by minimizing interconnection strengths between

distant synaptic elements to favor locality of computation. Additional problem-specific constraints could also be incorporated in the neuromorphic energy function [2]. But, in contradistinction to the traditional approaches, our methodology incorporates them directly into the trained (operational) network, as discussed in [1,3].

Lyapunov stability requires an energy-like function to be monotonically decreasing in time. Since in our model the internal dynamical parameters of interest are the synaptic strengths T_{ij} of the interconnection topology, the decay constants κ_i, the gain parameters γ_i and the Lagrange multipliers ${}^k\lambda_i$, this implies that

$$\dot{E} = \sum_i \sum_j \frac{\partial E}{\partial T_{ij}} \dot{T}_{ij} + \sum_i \frac{\partial E}{\partial \kappa_i} \dot{\kappa}_i + \sum_i \frac{\partial E}{\partial \gamma_i} \dot{\gamma}_i + \sum_l \sum_i \frac{\partial E}{\partial {}^l\lambda_i} {}^l\dot{\lambda}_i < 0 \quad (3.5)$$

One can always choose, with $\tau > 0$

$$\dot{T}_{ij} = -\tau_T \frac{\partial E}{\partial T_{ij}} \quad (3.6)$$

with similar expressions for $\dot{\kappa}$ and $\dot{\gamma}$, where τ_T introduces an adaptive parameter for learning to be specified in the sequel. Then, substituting in Eq. (3.5) and denoting by $\oplus$ tensor contraction, i.e., sum over all relevant indices, one obtains

$$\nabla_\lambda E \oplus \dot{\lambda} < \tau_T \nabla_T E \oplus \nabla_T E + \tau_\kappa \nabla_\kappa E \oplus \nabla_\kappa E + \tau_\gamma \nabla_\gamma E \oplus \nabla_\gamma E \quad (3.7)$$

Without loss of generality, one can assume $\tau = \tau_T = \tau_\kappa = \tau_\gamma$. The equations of motion for the Lagrange multipliers ${}^l\lambda_i$ must now be constructed in such a way that Eq. (3.7) be strictly satisfied. In addition, when the constraints are satisfied, i.e., ${}^l\tilde{I}_n \to 0$, we require that ${}^l\dot{\lambda}_i \to 0$. We have adopted the following analytical model for the evolution of λ,

$${}^l\dot{\lambda}_i = \tau \frac{\Pi}{\Lambda + 1/(\Lambda + \theta)} {}^l[\nabla_\lambda E]_i \quad (3.8)$$

where $\Pi = \nabla_T E \oplus \nabla_T E + \nabla_\kappa \oplus \nabla_\kappa E + \nabla_\gamma E \oplus \nabla_\gamma E$ and $\Lambda = \nabla_\lambda E \oplus \nabla_\lambda E$ and $0 < \theta \ll 1$. It is straightforward to prove that this model fulfills the above requirements.

4. Adaptive Learning using Adjoint Operator Theory

We now focus on the derivation of an algorithm for computing $\nabla_T E$ and $\nabla_\lambda E$. An adiabatic framework is assumed. On differentiating Eqs. (3.3) with respect to T_{ij} we get

$$\frac{\partial E}{\partial T_{ij}} = \omega_{ij} T_{ij} - \sum_k \sum_n {}^k\lambda_n \, {}^k\Gamma_n^{\alpha-1} \, {}^k\hat{\varphi}_n \frac{d}{dT_{ij}} {}^k\tilde{u}_n. \quad (4.1)$$

where ${}^k\hat{\varphi}_n$ denotes the derivative of the neural response. The computation of $\nabla_T E$ thus requires that N algebraic equations be solved for each parameter T_{ij}, that is N^3 equations at each iteration step. Similar requirements exist for evaluating $\nabla_\kappa E$ and $\nabla_\gamma E$, since one must obtain the values of $\frac{d^k\tilde{u}_n}{d\kappa_i}$ and $\frac{d^k\tilde{u}_n}{d\gamma_i}$ respectively. It is possible, however, to drastically reduce this computational complexity by introducing algorithms based on **Adjoint Operator theory** [4]. Specifically, let

$$\bar{p} = \{ T_{11}, \cdots T_{NN} \mid \kappa_1, \cdots \kappa_N \mid \gamma_1, \cdots \gamma_N \mid \cdots \} \quad (4.2)$$

denote the set of relevant parameters, i.e., $E = E(\bar{u}, \bar{p})$. Also, let

$$\bar{f}(\bar{u}, \bar{p}) = 0 \quad (4.3)$$

represent the nonlinear system equations. For notational simplicity we will omit the k–dependence. Then

$$\frac{dE}{dp_\mu} = \frac{\partial E}{\partial p_\mu} + \frac{\partial E}{\partial \bar{u}} \cdot \frac{\partial \bar{u}}{\partial p_\mu} \quad (4.4)$$

Since E is given analytically [3.3], the computation of $\frac{\partial E}{\partial p_\mu}$ and $\frac{\partial E}{\partial \bar{u}}$ is straightforward. To obtain $\frac{\partial \bar{u}}{\partial p_\mu}$ [e.g., $\frac{\partial^k \tilde{u}_n}{\partial T_{ij}}$ in Eq. (4.1)] we differentiate the state equations :

$$\frac{\partial \bar{f}}{\partial \bar{u}} \cdot \frac{\partial \bar{u}}{\partial p_\mu} = - \frac{\partial \bar{f}}{\partial p_\mu} \quad (4.5)$$

Denoting the NXN matrix $\frac{\partial \bar{f}}{\partial \bar{u}}$ by A, the vector $\frac{\partial \bar{u}}{\partial p_\mu}$ of unknowns by ${}^\mu\bar{z}$ and the "source" vector $-\partial \bar{f}/\partial p_\mu$ by ${}^\mu\bar{s}$, we can write

$$A \, {}^\mu\bar{z} = {}^\mu\bar{s}$$

As mentioned above, such a system of equations must be solved for each parameter μ, since the source term explicitly depends on μ.

Now, let A^* denote the formal adjoint of the operator A. Then by definition

$$A^* \, {}^\mu\bar{z}^* = {}^\mu\bar{s}^* \quad (4.7)$$

and

$$\begin{aligned} {}^\mu\bar{z}^* \cdot (A \, {}^\mu\bar{z}) &= {}^\mu\bar{z}^* \cdot {}^\mu\bar{s} \\ &= {}^\mu\bar{z} \cdot (A^* \, {}^\mu\bar{z}^*) \\ &= {}^\mu\bar{z} \cdot {}^\mu\bar{s}^* \end{aligned} \quad (4.8)$$

Recalling that

$$\frac{dE}{dp_\mu} = \frac{\partial E}{\partial p_\mu} + \frac{\partial E}{\partial \bar{u}} \cdot {}^\mu\bar{z} \quad (4.9)$$

and that

$${}^\mu\bar{s}^* \cdot {}^\mu\bar{z} = {}^\mu\bar{z}^* \cdot {}^\mu\bar{s} \quad (4.10)$$

we can identify

$$\frac{\partial E}{\partial \bar{u}} \equiv \bar{s}^* \tag{4.11}$$

Thus, the source term of the adjoint equations is independent of μ. Hence, the solution of a single set of adjoint equations will provide all the information required to compute the gradient of E with respect to all parameters. To underscore that fact we shall denote $\bar{z}^*$ as $\bar{v}$, to obtain

$$\frac{dE}{dp_\mu} = \frac{\partial E}{\partial p_\mu} + \bar{v} \cdot^{\mu} \bar{s} \tag{4.12}$$

In the above expression, the matrix A is defined as

$$^{k}A_{ni} = {}^{k}\eta_i \, \delta_{ni} - T_{ni} \, {}^{k}\hat{\varphi}_i \tag{4.13}$$

where

$$^{k}\eta_n = \begin{cases} \kappa + \frac{1}{3}[a_n^k - \varphi(^{k}\tilde{u}_n)]^{-2/3} \, {}^{k}\hat{\varphi}_n & \text{if } n \in S_X \cup S_Y \\ \kappa & \text{if } n \in S_H \end{cases} \tag{4.14}$$

The adjoint equations are then simply,

$$\dot{v}_i + {}^{k}\eta_i \, v_i = {}^{k}\hat{\varphi}_i \, [\sum_m T_{mi} \, v_m + {}^{k}\lambda_i \, {}^{k}\Gamma_i^{\alpha-1}] \tag{4.15}$$

In our previous work [1,3], the computation of $\frac{\partial^k \tilde{u}}{\partial T_{ij}}$ in (4.1), was based on a variant of the relaxation procedure suggested by Pineda [11]. The equilibrium points $^{k}\tilde{v}_i$ (obtained when, $\dot{v}_i \rightarrow 0$) are then used in the computation of $\nabla_T E$ in Eqs. (4.12). The *neural learning equations* can thus be expressed as

$$\dot{T}_{ij} = -\tau \, [\omega_{ij} \, T_{ij} - \sum_k {}^{k}\tilde{v}_i \, \varphi(^{k}\tilde{u}_j)] \tag{4.16}$$

So far the adaptive learning rate, i.e., τ in Eq. (3.6), has not been specified. The implications of its selection on the convergence of dynamical systems, have been discussed in our previous work [1,3,13]. We seek τ in the form

$$\tau \propto |\nabla E|^{-\beta} \tag{4.17}$$

where ∇E denotes the vector with components $\nabla_T E$, $\nabla_\kappa E$, $\nabla_\gamma E$ and $\nabla_\lambda E$. We have also shown that dynamical system, e.g., $\frac{d}{dt} |\nabla E| = -\chi |\nabla E|^{1-\beta}$ suffers a qualitative change for $\beta > 0$: it loses uniqueness of solution. The equilibrium point $|\nabla E| = 0$ becomes a singular solution being intersected by all the transients, and the Lipschitz condition is violated. By analogy with our previous results [1,3,13,14] we choose $\beta = 2/3$, which yields

$$\tau = \Big(\sum_n \sum_m [\nabla_T E]_{nm}^2 + \sum_i [\nabla_\kappa E]_i^2 + \sum_i [\nabla_\gamma E]_i^2 + \sum_k \sum_i {}^{k}[\nabla_\lambda E]_i^2 \Big)^{-\frac{1}{3}} \tag{4.18}$$

Neural learning methodologies based on dynamical systems must consider the fundamental problem of network stability as synaptic weights evolve during training. Previously published approaches [1,8,10,11] ignored the issue. To make some progress, we observe that if each node of the activation network had an associated terminal attractor , the resulting neurodynamics would be unconditionally stable. However, since by definition of a mapping, data are provided only for neurons in the input and output topographic partitions, "virtual" attractors must be determined for the hidden units. These attractors are virtual since they correspond to a current estimate of the synaptic connectivity matrix. Specifically, Eq. (2.2) has to be modified to read

$$^{k}I_n = \begin{cases} [^{k}a_n - \varphi(u_n)]^{\beta} & \text{if } n \in S_X \cup S_Y \\ [^{k}z_n - \varphi(u_n)]^{\beta} & \text{if } n \in S_H \end{cases} \tag{4.19}$$

where the virtual attractor coordinates z_n are obtained by considering the fixed point equations (3.2), as adaptive conservation equations which utilize the extra degrees of freedom made available by the hidden neurons in S_H. Thus, if we define

$$^{k}\tilde{H}_n = \sum_{l \in S_X \cup S_Y} T_{nl} \, \varphi(^{k}a_l) \tag{4.20}$$

we can compute the virtual attractors (in the terminal attractor formalism) from

$$\dot{z}_n = -[\kappa_n z_n - \sum_{m \in S_H} T_{nm} \, \varphi(z_m) - {}^{k}\tilde{H}_n]^{1/3} \tag{4.20}$$

Note that the above expression also involves terminal attractor dynamics.

5. Operational Network

The activation dynamics of the operational network can be used to directly encode application-specific as well as event-driven constraints. This is a significant departure from the existing neuromorphic models for constraint satisfaction. Two modes for constraint handling had hitherto been adopted [1]: generate the training samples in conformity to a prespecified criterion, or incorporate them into the neuromorphic objective function given by Eqn. (3.3) [2]. Both these approaches are restrictive and computationally inefficient. Since the operational scenario needs to be known "a priori", they *skew network behavior*. A neural network so trained learns only limited (presented) aspects of the nonlinear functional invariant. Each additional constraint requires retraining the network, thereby severely limiting real-time performance.

In this perspective, we propose an "a posteriori" regularization methodology that precludes modification of the synaptic configuration. Our methodology entails adding constraint information into the dynamics of the

trained network along concept lines inspired by "renormalization theory", a methodology widely used in physics. The activation dynamics in the operational mode then takes the following form:

$$\dot{u}_n + \kappa_n u_n = \sum_m W_{nm}\varphi_\gamma(u_m) + I_n + \sum_r {}^r\lambda_n[\nabla_{\bar{u}} g_r(\rho_r, \bar{u})] \tag{5.1}$$

where W_{nm} denotes the learned interconnection matrix, and the "working" input source I_n is now defined as,

$$I_n = \begin{cases} [x_n - \varphi(u_n)]^{-1/3} & \text{if } n \in S_X \\ 0 & \text{if } n \in S_H \cup S_Y \end{cases} \tag{5.2}$$

The constraints $g_r(\cdot)$ relate to application-specific considerations, while ρ_r denote constraint-specific renormalization parameters which scale the neuron activities, constructed so that the gradient contribution to the dynamics vanishes whenever the actual constraint is satisfied. An illustrative example on deriving analytical expressions and the concomitant renormalization parameter, in the context of learning nonlinear mappings for robotic control, is presented in [1].

6. Simulation Results

The computational framework developed in the preceding section has been applied to a number of problems, including signal reconstruction [6] and robotics, e.g., inverse kinematics of redundant manipulators [1,2], that involve learning nonlinear mappings. In the sequel we discuss only one of our experiments which involved learning a continuous clipping nonlinearity. This function has been extensively benchmarked [9], and provides an adequate basis for illustrating the computation efficacy of our proposed formulation as compared to the existing neural learning algorithms. The test setup included a fully connected network with one neuron in the input and output sets respectively, and two hidden neurons. The training set consisted of 12 training samples.

Since our learning methodology involves singular solutions of highly coupled, continuous dynamical systems, extreme caution must be exercised when simulating the algorithms in a digital computing environment. For example, explicit methods such as Euler or Runge-Kutta cannot be used, since the presence of singularities induces extreme stiffness. Practically this would require an integration time-step of infinitesimal size, resulting in numerical round-off errors of unacceptable magnitude. Clearly, fully implicit integration techniques have to be used [3]. For the simulations reported below, Eqs (2.1. and 2.3.11) are integrated using a fourth-order Kaps-Rentrop scheme. Figure 1(a) shows the LMS error during the training phase. The worst-case convergence of the output state neuron to the presented attractor is displayed in Fig. 1(b). Note that the system learns the nonlinear map **orders of magnitude faster** than by the back-propagation algorithm (0.2 secs vs 5000 secs effective time). The network yielded an interpolation/recall precision with error under 0.3% for 80% of the test samples. The worst-case error detected was 3.5%. In contrast, a multi-layered perceptron network (configuration defined in [9]) and trained using the backpropagation algorithm required 5000 seconds effective time to achieve a worst-case error of 6%.

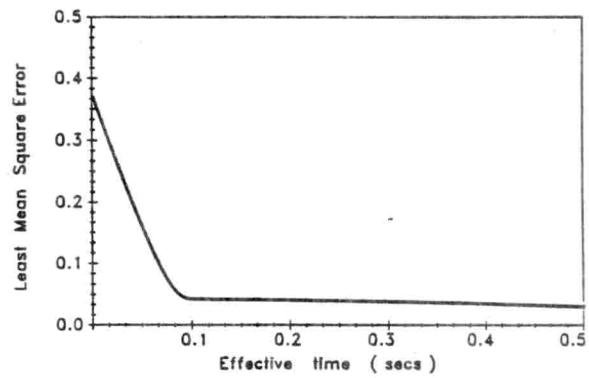

Figure 1(a) Profile of the least-mean-square (LMS) error during the learning phase

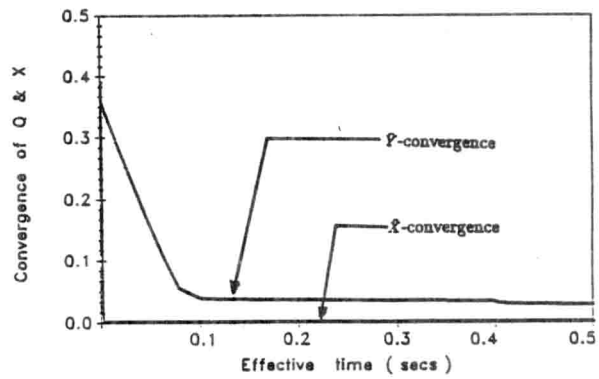

Figure 1(b) Worst case convergence of the input and output state topographic partitions to the presented attractors

7. Conclusions

In this paper we have presented a new theoretical framework for learning continuous nonlinear mappings using artificial neural networks. Central to our approach are the concepts of terminal attractors - a new class of mathematical constructs which provide unique information processing capabilities to the artificial neural system and *adjoint operator theory* which enables an efficient computation of energy function gradients. The rapid network convergence resulting from the infinite local stability of terminal attractors enables the development of very

fast neural learning algorithms, an essential requirement for applications in adaptive signal processing and robotic control. Our methodology is implemented on topographically partitioned, but densely connected networks, to facilitate the encoding of training samples as terminal attractors. In a significant departure from prior neuromorphic formulations, we completely decouple the incorporation of applicational or event-driven constraints from the network learning phase by introducing a renormalization mechanism. Thus, initially the network is trained using pairs arbitrarily sampled over the entire function domain. Then, during run-time, the desired constraints are included in the dynamics such that network convergence necessarily ensures constraint satisfaction. Notice that this method requires no additional training. Finally, by introducing virtual terminal attractors we guarantee the unconditional stability of the activation dynamics during learning.

Acknowledgements

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology. Support for the work came from agencies of the U.S. Department of Defense, including the Innovative Science and Technology Office of the Strategic Defense Initiative Organization. We wish to acknowledge fruitful discussions with M. Zak and F. Pineda.

References

[1] J. Barhen, S. Gulati and M. Zak, "Neural Learning of Constrained Nonlinear Transformations", *IEEE Computer*, 22(6), 1989, pp. 67-76.

[2] J. Barhen and S. Gulati, "Self-Organizing Neural Architecture for Inverse Kinematics of Redundant Manipulators", *NATO ASI*, F44, 1989 (in press).

[3] J. Barhen, M. Zak and S. Gulati, " Fast Neural Learning Algorithms Using Networks with Non-Lipschitzian Dynamics", in *Proc. Neuro-Nimes '89* , Nimes, France, Nov. 13-16, 1989 (in press).

[4] J. Barhen et al., "The Application of Adjoint sensitivity Theory to a Liquid Fuels Supply Model", *Energy*, 9(3), 1984, 239-253.

[5] S. Grossberg and M. Kuperstein, "Neural Dynamics of Adaptive Sensory-Motor Control", North Holland, 2nd Edition, 1989.

[6] S. Gulati and J. Barhen, "Predictive Learning Algorithms for Multidimensional Signal Reconstruction", in *Third Annual Parallel Processing Symposium*, at Fullerton, CA, March 29-31, 1989.

[7] J.J. Hopfield "Neurons with Graded Response have Collective Computational Properties Like Those of Two-State Neurons," *Proc. of Nat'l Academy of Sciences*, 81, 1984, 3058-3092.

[8] A. Lapedes and R. Farber, "A Self-Optimizing, Nonsymmetrical Neural Net for Content Addressable Memory and Pattern Recognition", *Physica D*, 22, 1987, 247-259.

[9] R.P. Lippmann and P. Beckman, "Adaptive Neural Net Preprocessing for Signal Detection in Non-Gaussian Noise", D.S. Touretzky, editor, *Neural Information Processing Systems*, New York, 1988, pp. 124-132.

[10] B.A. Pearlmutter, "Learning State Space Trajectories in Recurrent Neural Networks: A Preliminary Report", *Tech. Rep. AIP-54*, Pittsburgh: Carnegie Mellon University, Dept. of Computer Science, 1988.

[11] F.J. Pineda, "Dynamics and Architecture in Neural Computation" *Journal of Complexity*, 4, 1988, 216-245.

[12] R.J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks", Neural Computation, 1, Spring 1989.

[13] M. Zak, "Terminal Attractors for Addressable Memory in Neural Networks," *Physics Letters A*, 133, 1988, pp. 218-222.

[14] M. Zak, "The Least Constraint Principle for Learning in Neurodynamics", *Physics Letters A*, 135, 1989, 25-28.

ASYNCHRONOUS NONLINEAR HOLONOMIC SIMULATION:
METHODOLOGIES IN MODELING INTERNATIONAL RELATIONS

Elizabeth K. Newton

Department of Political Science
University of California, Berkeley
Berkeley, California 94720

Phillip Alvelda

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

ABSTRACT

We present a radical re-conceptualization of the cognitive processes (information processing and learning) performed by social systems. Using artificial neural network theories, the focus of international relations theory switches from the power and attributes of actors, to process and connection between actors. The new model thereby provides a new methodology for understanding learning in collectives and the role of knowledge and information flow in cooperative endeavors. Major advantages to this new approach also lie in the clarification of collective learning, not just qualitatively, but, quantitatively as well, with the newly enabled use of whole classes of mathematical tools such as statistics, thermodynamics, constrained optimization theory, and nonlinear dynamics. The proposed neural model proves compatible with other accounts of international cooperation, yet goes beyond previous economic and cybernetic models by actually accounting for "concurrently asynchronous" collective processing.

1. INTRODUCTION

> Politics, like all techniques of making and implementing decisions, is not an end in itself...politics in the world today is an essential instrument of social learning....a process of awakening [1].

In his far-seeing work, Karl Deutsch grasped the importance of "learning" and information flow as a means for fostering peaceful interaction, stability and cohesion of national governments, states, and people. In international relations theory, "learning" comprises one of the most tantalizing and, at the same time, frustrating concepts: Can states learn in order to cooperate and maintain peace, and if so, how? In a human being, individual learning occurs at several levels, from the biochemical adjustment of neuronal synapses, to the development of neural reflex arcs, to complex and abstract philosophical realizations. But can the same concepts of learning be applied at even higher levels, to those of collective social organizations, states, and regimes?

2. REGIME THEORY

Before delving into new analogies and methodologies, however, the definition of international regimes and their place in international relations theory is warranted. A regime regulates or influences states or international actors' behavior for the pursuit of some common goal. For example, regimes have developed to address problems such as uncertainty in international financial markets, use of the seas, and barriers to free trade. International regimes have been variably described as norms, principles, or rules and procedures held collectively for the purpose of addressing problems in specific issue areas. Often regimes are envisaged as actual entities, multilateral agreements, or institutions, although this is not necessarily the case: it is merely conceptually easier to consider them in this formalistic way. So, a regime is a problem-solving effort resting on a consensus of knowledge among the actors, which evolves in response to changing national needs as conditioned by ideology and consentual knowledge.

In light of regimes' problem-solving nature, naturally the question arises as to how regimes handle or store information and learn. Most generally, learning entails the acquisition of knowledge. In a regime, learning is held to constitute the convergence of actors' knowledge which serves as a guide to policy. Learning then is an evolutionary process motored by successive experiments on the part of the actors. For the collective learning process to occur, there must be some sort of institutional memory which enables the remembrance of particular parameters and already-attempted solutions. Therefore collective memory (as distinct from individual memory) must be defined. Additionally, a mechanism for learning must be specified. How do collectives arrive at correlations between the past and desired future outcomes? And finally, indicators of the process must be available, preferably ones which do not depend solely on empirical observation of performance, but rather are derived from the process itself.

3. PREVIOUS MODELS

Problems abound when "learning" is used to describe the behavior of collectives without sufficient justification or theoretical self-consciousness. One of the reasons that political theorists have had so much trouble achieving useful, repeatable results is because the computational and cybernetic models of cognitive sys-

tems on which they base their work, utilizes a type of information processing and memory that inherently limits their computational modeling capability.

A typical example can be found in the decision-making and collective behavior theories which use linear algebraic or differential equations. Such models can only incorporate a small number of linear variables and therefore model more complex systems rather poorly. The system dynamics are described by several linear equations, as opposed to the coupled non-linear differential equations, that are absolutely necessary to model non-linear phenomena. Most current political system models cannot begin to approximate the numerical complexities of nonlinear reality; with any number of nonlinear terms, the generalization from linearized approximation analysis to actual nonlinear system behavior is typically invalid [4]. Linear system models cannot exhibit the behavioral complexities that exist in even fairly simple nonlinear systems; characteristics such as extreme sensitivity to initial conditions behavioral bifurcation, and deterministic chaos are typical only in nonlinear systems. With the simplified linear models of nonlinear systems as a rather weak foundation, theories that use descriptions of learning in order to explain system evolution typically resort to incomplete or static black-box definitions.

For reasons such as these, older social system models cannot reliably emulate realistically complex, coupled systems that involve the processing of large amounts of data under uncertain conditions. These models all mistakenly neglect the effect of distributed memory, and assume that collectives arrive at conclusions in much the same way as one individual performing a single task.

The proposed neural network methodology can describe 'social' collective behavior, taking into account the non-linear functional characteristics of both individuals and the previously unrealized importance of their interactions, as well as 'noisy' or inadequate information flows. Such highly parallel, nonlinear computational systems mimic much more closely the cognitive processes of social collectives and regimes, and provide rigorous mathematical methods for modeling collective social learning.

4. NEW ANALOGIES

Collective problem-solving, is comprised of many individuals working toward a solution at the same time, providing, as one would hope, a net result greater than the sum of its parts. People function in a decidedly nonlinear fashion, basing their decisions on a plethora of visual, aural, and tactile sensory information. The particular information sources that a person selects for decision formulation purposes, directly influences the amount, quality, and bias of that received information (i.e. which opinionated lobby group gets to a politician first can have marked influence on voting behavior). Stored information that governs social system behavior is embedded not only in the characteristics of an individual actor, but also in the pattern, quantity, and quality of his interactions and relations with others.

Each must handle large quantities of information and attempt correlations in order to achieve appropriate outcomes. On a smaller scale, parallels can be drawn between the units which compose both regimes and neural networks. States, or international actors are held together, much as neurons are, by an organizing, hierarchical principle. Regimes bring individual units together to operate upon a task; so, like neurons, states may be considered processors, albeit fairly complex ones, which work in parallel on a particular task defined by the regime. The analogy can easily be extended to note that complex state-neurons can also be modelled by sub-networks like those unifying different governmental bureaucracies, local offices, people, etc. Social paths of information are the equivalents of neural interconnections which may vary in importance or influence (weight), number (depending on the number and topology of units). Information is stored collectively in the communication between units, and collective computation is performed on the basis of this distributed information.

Regime components also have their representation in neural networks. First of all, the norms on which the regime rests may be considered analogous to the desired outcomes presented to the network. The norms of a regime tend to be articulations of overlapping "national interests." Similarly, the regime's rules and procedures correspond to limitations on the network's interconnection weights and patterns since they dictate the constraints to available solution configurations necessary to achieve intended outcomes. Rules are required to prevent 'jamming' and to prioritize the arrangement of channels that states use to communicate with each other.

In addition to general information storage and handling similarities, learning in regimes involves changes in knowledge such that the difference between the desired end and the actual outcome are reduced. That is, learning occurs with changing connectivity (amount and accuracy of information flow, i.e. level and flavour of negotiations) between states. When the difference between actual outcome and the desired end is great, then the institution re-attempts a solution, altering as needed the way in which, and from whom, it aggregates and processes its knowledge. This error minimization process is not dependent on the ability of any one unit to perform better or worse; it relies on the regime's collective capacity to manipulate, coordinate and apply global activity. The learning and adaptation process is on- going and can result in widely varies system behavior: from globally stable systems to systems with many locally stable points, to systems that oscillate in limit cycles, to systems that evolve chaotically. Overall, this range of possible descriptions fits nicely with observed regime evolution. Learning is then seen as the linking and delinking of issues is a continuously evolving dynamical trial and error processes. Fundamentally, negotiation or linkage, as a consequence of feedback, characterizes regime processes.

The behavior of more simple static networks (i.e. Backwards Error Propagation [7]) that lack high plasticity, that is, an ability to adapt to or integrate new, unexpected inputs after a training cycle has been completed, exhibit the same behavior in regimes that are

labeled "lags." A regime is unable to immediately alter its connectivity to change its solution, whether due to habit, uncertainty, or cognitive failing (inability to perceive alternative solutions). At root, this quality stems from the networks' lack of a dynamic control structure. Established actors in a given regime would have regular official and social contacts with whom they interact in order to most efficiently (in the case of the ideal politician) accomplish their inter-regime tasks. When an entirely new environmental input becomes significantly relevant, new channels, advisors, funding sources, etc... must be found and integrated to sufficiently support new tasks and duties. So, while slight changes in the environment can be handled by generalizations from previous solutions, abrupt large-scale changes require a modifiable or dynamic network hierarchy.

Additionally, the neural model does not even presume or require the rationality of the individual policymaker for the proper function of a regime as a whole. If one unit processes poorly(i.e. a single policy maker is not very good at his job), information can be redistributed by the network so that a different solution is achieved. Regimes can learn even though certain individuals composing it may be inefficient, incompetent, or absent. This is because a large number of individual elements contribute to any single global result. Small changes in the input, or even deletions of individual elements of a network, produce very little change in the global output. In this way, the network is very fault-tolerant.

One criticism levelled by Strange [8] is that regime theory is narrowly rooted in state-centric analysis. Yet neural network modularity characteristics permit the unit level of analysis to vary from that of individuals through larger and larger associations such as states, nations, and multi-nationals like NATO. The networks are modular in the sense that large super-networks can be constructed from densely interconnected sub-networks. In this way large hierarchical systems with different scales of complexity and data availability can be easily simulated at different levels of analysis.

Strange [8] also articulated that current regime theory suffers from an overemphasis of the static, concrete elements. Because 'regime' implies order, analyses show a bias toward the status quo and unchanging procedures. Regimes, however, are dynamic and evolve over time, sometimes with great discontinuities. Factors contributing to change, like new information and changing interpretations of "national interest" are often overlooked. But, the neural network model accounts for changes within regime hierarchies(changing interconnectivities, or rules and procedures) as well as changes to individual regime actors themselves (the incorporation of new actors, death of old ones), and error-functional adaptation, (or the change in norms or goals in an autonomous regime).

In contrast to older regime models, the more complex neural theories use dynamical system models consisting of coupled differential equations. In other words, without artificial constructs, they can evolve in time just as the real-world systems they model with a complete range of possible system behaviors from stability to chaos. A typical example is a fully connected, additive-type neural system (e.g. a Hopfield model) defined by the following system of coupled differential equations:

$$\dot{u}_i + a_i u_i = \sum_j T_{ij} g_j(\gamma_j u_j) + I_i. \qquad (4.1)$$

Here u_i represents the internal state of the $i-$th neuron, T_{ij} denotes the synaptic coupling from the $j-$th to the $i-th$ neuron and I_i is the external input bias. The sigmoidal function g_j modulates the neural response, γ_j denotes the transfer function gain for the $j-$th neuron and a_i represents the inverse of a characterisitic time constant or the decay scaling term. Euler's approximation to the above system of continuous-time equations

$$\dot{u}_i = (u_i^{t+1} - u_i^t) \,/\, \Delta$$

where Δ denotes the discretization stepsize, defines the Hopfield operator

$$\varphi_i(\bar{u}) = u_i + \Delta\left[- a_i u_i + \sum_j T_{ij}\ g_j(\gamma_j u_j) + I_i \right] \qquad (4.2)$$

How does one describe this neural model's operation in real-life terms? Imagine a regime made up of many representatives of separate states, each in communication with the others. Their task demands that they work together, to generate optimal formulas for achieving the desired end. The number of inputs that each individual must consider is enormous. They include domestic public opinion, budget constraints, memories of past performances, goals external to the immediate task at hand, personal values, diplomatic, social and cultural constraints, each of which can also introduce further complexity in terms of performance evaluation feedback. Whether or not the network succeeds depends on the group's ability to synthesize the inputs, communicate the decisions (outputs) to others, and build a coalition agreeing on a particular collective path to achieve a goal. Choices of the proper channels depends on the realization of specific interrelationships between past, present, and future experiences, as well as the history and projected future relations between individual and group capabilities.

5. AN ASYNCHRONOUS NEURAL FORMALISM

A neural network model of an existing regime is in progress (Alvelda and Newton [10]) in which a large number of environmental inputs are presented to the network, so the model is not intrinsically biased in its range of possible solutions. The weights that evolve in the various interconnections are examined to discern which actors might be most vital to the achievement of particular outcomes and how they are coupled. In other words the network is allowed to evolve under characteristic internal dynamics in order to develop its own internal representation of the critical invariant properties necessary to accomplish specified regime functions. Analysis of these emergent properties promises to reveal factors particularly pertinent to policy development and decision-making, and provide a model that is closely analogous to the functional methodologies of existing social systems and regimes.

The model is based upon a mathematical formalism from Barhen and Gulati [9], a new neural algorithm that incorporates a system of closely coupled nonlinear differential equations that can simulate a distributed system which evolves in a "concurrently asynchronous" fashion.

We now briefly summarize some key attributes of concurrently asynchronous neural computation [9] to highlight the divergence of older social system models from actual regime behavior. Concurrent tasks are accomplished most efficiently by a collection of functionally cooperating processes with no explicit time interdependencies that require local waiting for the purpose of swapping partially computed results. Computation is then essentially iterative in nature, with system dynamics controlled by asynchronously updated state variables and previous time history.

This asynchronous methodology provides an effective tool for designing system models capable of delivering high throughput since synchronization and coordination restrictions are eliminated. Computations can be carried out without adherence to precedence constraints, allowing each individual processor to operate at its own best speed on **available** (albeit incomplete) data. In a neural network implementation, a neuron would be allowed to fire without having to wait for output signals from all the other neurons to which it is connected.

Asynchronous dynamics also imply a new type of local fault tolerance. Individual neurons can remain idle for finite periods without significantly altering the global computation. This is closely analogous to the "refractory" or recharging periods observed in biological neurons. In terms of system simulation, the elimination of time-constrained inter-neuron dependence, allows immediate information rerouting around failed network segments and the continuation of processing without global reconfiguration. Another advantage is the potential for simulation of heterogeneous collective ensembles (i.e. systems in which different processing nodes have different performance characteristics) in order to achieve hierarchical neuronal processing.

Concurrently asynchronous updating can be described as follows: at some operating instant t, an idle neuron, i, initiates the update of its state to

$$u_i^t = \varphi_i(\bar{u}_{N-\{i\}}^{t-1})$$

using state information already available from the previous updates, without waiting to receive the results of still ongoing activations, where $\bar{\varphi}_i^t$ is the ith component of the Hopfield operator defined in (4.2). The precise strategy for selecting available state information would depend on the degree of synchrony and external control in the system being modeled.

Typical behavior for this type of network with asynchronous feedback was found to be deterministic chaos [9]. Useful behavior, on the other hand, is typified by the uniform contraction in phase space of independent neurons in convergence toward a single global solution. It is not until constraints were placed upon the parameters and control structure of the network that sufficient conditions exist for the network to converge to a solutions notwithstanding the bounded time delays and inefficiencies of the system. In summary, concurrently asynchronous dynamics implicitly define attributes that are essential for accurate neurocomputational modeling and simulation of social, political, and economic systems.

6. DISCUSSION

Given the similarities of regimes and neural networks, several conclusions can be drawn from these new connectionist cognitive models to describe the operation and evolution of social systems and international regimes.

Obviously, change occurs when there is a large influx of new knowledge (environmental change or a substantial alteration to the general fabric which comprises the network as a whole). New information, cooperation and/or support, can enable new solutions to old problems confronting states. But learning also occurs between cataclysmic environmental changes. People(neurons) continuously adjust the way and with whom they interact in an effort to improve performance. When feedback information indicates that global performance is high, the magnitude of adaptation is small. When catastrophic change impinges upon the collective, large scale changes and reorganizations must occur before the regime can once more operate effectively. The system learns continuously.

Whether or not a distributed system can eventually learn to solve a given problem depends on many factors in addition to the net resources of all the individual components. The way in which those individuals are managed must take into account the characteristic dynamical properties of: individuals, the interactions between them, processing and communication inefficiencies, and task decomposition methodologies. External constraints must be imposed to gaurantee constructive collective function in spite of inherhent uncertainty, asynchrony and individual failures.

Since knowledge is diffused throughout the network, the model also explains how actors come to integrate knowledge into a coherent whole and understand interrelationships and preserve it over time even though individual careers end and replacements are initially untrained and may ultimately function very differently. In this way the model emphasizes the importance of knowledge and cognitive change in collective negotiation and the emergence of relational cooperation.

While neural regimes seem to elucidate what could be called social learning, the model also points to reasons that might explain the collapse of regimes. When a regime lacks sufficient external connectivity (the quality and quantity of environmental information) and topological complexity (scale, variety, and complexity of interrelation and information processing capabilities), then its tendency to fail in a given task will be all the greater since it cannot sufficiently model or even approximate a solution. In regimes with many actors, information dysfunctions and failures in individual units then become less important than the quality of communication between them. This finding echoes one of Deutsch's [3] perceptions from 25 years earlier when he argued that self-closure (insufficient input) poses a major threat to the continuation of political organizations.

Ultimately, neural network theories provide wonderfully rich models for describing the functioning of international regimes. The model substantiates the idea of higher order, collective cognition. International memory resides in inter-state relations. Regimes can learn, provided their components associate with one another in an interdependent manner.

References

[1] G.M. Baudet, "Asynchronous Iterative Methods for Multiprocessors," Journal of ACM, Vol. 25(2), pp. 226-244, 1978.

[2] D. Chazan and W. Miranker, "Chaotic Relaxations," Linear Algebra and its Applications , Vol. 2, pp. 199-222, 1969.

[3] K. Deutsch, The Nerves of Government, NY: Free Press, 1963.

[4] Guckenheimer and Holmes, Nonlinear Oscillations, Dynamics, and Bifurcations of Vector Fields [Springer Verlag, New York, 1983]

[5] H.T. Kung, "Synchronized and Asynchronized Parallel Algorithms for Multiprocessors," Algorithms and Complexity: New Directions and Results, J.F. Traub, Ed. , Academic Press, New York, pp. 428-464, 1976.

[6] C.M. Marcus and R.M. Westervelt," Stability of Analog Neural Networks with Delay," Physical Review A , (in press).

[7] D. E. Rumelhart and J. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, : Vol. 1: Foundations, Cambridge: MIT Press, 1986.

[8] Strange, Susan. "Caves! Hic Dragones: A Critique of Regime Analysis," International Organization, 41(4), Autumn 1987.

[9] J. Barhen and S. Gulati, "Chaotic Relaxation in Concurrently Asynchronous Neurodynamics," (submitted to Proc. of 1989 Int'l Joint Conf. Neural Networks at Washington, D.C.).

[10] P. Alvelda and E. Newton, "Event Driven Asynchronous Political Simulation," (in preparation).

DNA: A New ASOCS Model With Improved Implementation Potential

George L. Rudolph and Tony R. Martinez
Department of Computer Science
Brigham Young University
Provo, Utah 84602

Abstract. A new class of high-speed, self-adaptive, massively parallel computing models called ASOCS (Adaptive Self-Organizing Concurrent Systems) has been proposed. Current analysis suggests that there may be problems implementing ASOCS models in VLSI using the hierarchical network structures originally proposed. The problems are not inherent in the models, but rather in the technology used to implement them. This has led to the development of a new ASOCS model called DNA (Discriminant-Node ASOCS) that does not depend on a hierarchical node structure for success. Three areas of the DNA model are briefly discussed in this paper: DNA's flexible nodes, how DNA overcomes problems other models have allocating unused nodes, and how DNA operates during processing and learning.

Introduction. A new class of high-speed, massively parallel, adaptive computational models called ASOCS (Adaptive Self-Organizing Concurrent System) has been proposed [3]. These present neural networks with fast, well-bounded learning algorithms using simple, asynchronous digital nodes [4] - [7]. One area of current ASOCS research focusses on building VLSI implementations of ASOCS using current technology. Chang and Vidal describe one possible implementation of an ASOCS model in [1]. More recent analysis of ASOCS shows that current ASOCS models may make inefficient use of limited network resources. This is not due to a problem inherent in the models, but in VLSI implementation. This paper introduces a new ASOCS model, called DNA (Discriminant-Node ASOCS), that is designed to overcome these inefficiencies. Some basic ASOCS terms are defined, and ideas briefly discussed. An example of how ASOCS models learn is presented, leading into a discussion of the problems of inefficient use of nodes in a hierarchical implementation of a network. The section following that discusses how DNA overcomes these problems. The paper concludes with a brief summary of DNA's advantages and limitations, possible applications for DNA and finally, the direction of current research.

Definitions and Ideas. An ASOCS network can be viewed as a function machine that can compute a mapping of arbitrary Boolean inputs to arbitrary Boolean outputs. A network is composed of computing nodes that operate asynchronously and in parallel. ASOCS networks have two modes of operation--processing and learning. During processing, the network acts like a parallel hardware circuit. During learning, nodes may change the functions they compute, or be deleted from the network altogether, as new information is presented. The network adapts itself so that the overall function it computes is consistent with the information it has received. The network learns **incrementally**, using rules that express a particular functional mapping. A given mapping or rule is known as an **instance**. The following are examples of instances:

AB'C -> Z ACD -> Z' ED' -> Y'

The variables on the left side of the arrow represent the inputs, and those on the right side the desired outputs for those inputs. Instances may specify high or low outputs for one or more output variables. An instance has **positive polarity** for a given output variable if it specifies that variable to be high, and **negative polarity** for that output variable if the variable is specified to be low. Nodes that specify opposite polarities for the same output variable are called **discordant** instances. In the examples above, AB' -> Z is positive for Z, while ACD -> Z' is negative for Z. These instances are also discordant instances. ED' -> Y is positive for X. It is not discordant with either of the other two instances because the specified outputs are different variables. For simplicity, the instances presented in the discussion will be limited to a single output variable (i.e. all outputs will be Z or Z' only). Implementing multiple outputs is an easy extension of the ideas presented here [3], but will not be addressed.

The collection of instances presented to the network is the **instance set**. The instance set is not stored explicitly in most models. Rather, the network will store some internal representation that allows it to execute the same function as the particular instance set it has been given. It has been shown that the number of instance sets that potentially represent the same functional information is infinite, and that therefore instance sets are not unique [3]. Furthermore, internal representations for the same instance set are diferent for different models. A network is said to **fulfill** a given instance set when the function it computes matches that specified by tthat instance set. Conflicts occur when a set of inputs could cause the network to output both high and low for some variable at the same time. The network resolves conflicts (is made **consistent**) by adding new nodes and variables to the network or deleting old conflicting ones. Precedence is given to the most recent instance presented to the network which is referred to as the **new instance** (NI).

Whenever a NI is presented to the network, each node *independently* determines its relationship to the NI. The nodes act in parallel, *without* information from any other nodes. This means they can also act asynchronously. Nodes take different adaptive actions based on their relationships to the NI. Each node, and by extension the network, reconfigures without control external to the network, the network is **self-organizing**.

After enough variables have been added so that the instance set is consistent with the NI, the network can enter the **Self-Deletion** (SD) phase. Nodes and variables that are *no longer needed* in discriminating inputs are deleted. The network does not lose information during SD, and will work properly without doing SD. Whereas consistency is absolutely necessary, SD is optional. SD allows the network to free nodes, an important factor when resources are limited (as in any physical implementation).

Learning in ASOCS. What follows is a brief example of the learning paradigm used in current ASOCS models. This is a brief overview of the process, presented to help the reader understand the sections that follow. This discussion deals only with instances and instance sets, not with particular models or networks, because internal representations differ from model to model. (It should be noted however, that as the "instance set" changes, a network must reconfigure to reflect those changes.) Assume the following instance set:

AB' -> Z

Then, the instance AB'CD -> Z' is added. There would be a conflict between the NI and the instance set when AB'.is matched To resolve the conflict, variables C and D must be added to the instance set. When this is completed, the instance set above will be replaced by the following one:

AB'C' -> Z
AB'D' -> Z
AB'CD -> Z'

During SD all, or only parts of some instances may be eliminated from the instance set. Two simple cases are given to illustrate this (by no means an exhaustive explanation). Suppose the following two instances are contained in some instance set:

ABC -> Z
ABCD -> Z

The second can be deleted because that case is covered by the first one. Consider the instances:

AB' -> Z

AB ->Z

These two can be replaced by:

A ->Z

which covers the same cases with less variables (and instances).

Once consistency is established, the instance set fulfills the NI asnd all parts of previous instances that do not contradict the NI. Once SD is complete, the instance set may contain fewer variables and instances than before. The NI presentation, comparison, adaptive actions and possible self-deletions occur *only once* for each NI. No iterative loops are necessary to obtain the correct instance set. This is reflected in the networks in that no iterative training loops are necessary to cause a network to learn properly.

Problem: Implementing Current Models in VLSI. The basic structure of a node in current models has two parts: a two-input logic gate capable of performing the 16 possible Boolean functions on two variables, and a control unit that tells it which function it is to perform at any given time. The node structure is illustrated in figure 1. Current models use hierarchies of the two-input nodes to represent instances in the current instance set. The problem that occurs is not a problem inherent in using the hierarchical structures in these models. Rather it is a problem of implementing the hierarchy in VLSI: the technology does not fit the models.

Recall that instances can have positive or negative polarities. Current models use two types of nodes to represent instances. The first type is called a **primitive node**, or **P-Node**. These "building-block" nodes have no real polarity, but compute their functions and feed their output to other nodes Figure 2 shows a representation of a P-Node, which has the function it computes written in, but no letter at the top representing the output and polarity. (Representing the nodes this way does not imply that this information is stored in the node itself--it may not be.) The other type is called a **discriminant node** or **D-Node**. Figure 2 also shows a D-Node, which has Z or Z' (Z bar) at the top, indicating the output variable and its polarity in addition to showing the function it computes In DNA, there are no "building-block" nodes. Every node is a D-Node. This distinguishes it from all previous models. Even though DNA is different, it still uses the ASOCS learning paradigm. Hence the name "Discriminant-Node ASOCS", or DNA.

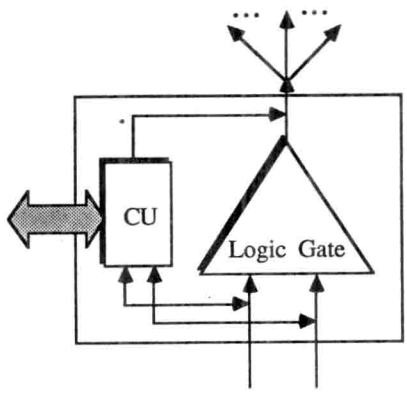

Fig. 1 - General Structure of ASOCS node

Both dynamic and static interconnect schemes [2] are found among current ASOCS models [3]. In considering a VLSI implementation of these hierarchical models, a problem associated with allocating unused nodes efficiently arises because the technology does not fit the models. Figure 3 shows how the instance set:

AB'C' ->Z

AB'D' ->Z

AB'CD ->Z'

might be represented using a dynamic scheme, which uses a router to change and set the links between nodes.

Implementing dynamic interconnect, routing through or around other nodes will become a problem as the number of allocated nodes in the network increases. A static interconnect is also preferred because it would be cheaper and simpler to implement. Figure 4 shows the

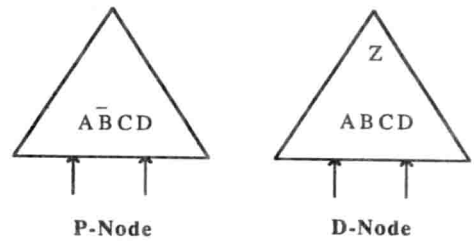

Fig. 2 - Representaion of an ASOCS P-Node and D-Node

instance set above represented using a static scheme. This one is a Binary Decision Tree [11] in which the links between nodes are fixed.

The problem is that it is not possible to decide beforehand which physical branch will require more nodes than the others, and thus which nodes will remain unused. In both the dynamic and static cases, allocating nodes from different parts of the network complicates the logic of the network. Each node represents only a small piece of some instance in the current instance set. Different pieces, i.e. nodes, have to be logically ordered so that the instance set is fulfilled. The physical position of a node in the network thus becomes important to the success of the network. Where the unused nodes are has an impact on how easily they can be allocated by a logical chain that may need them. It is the hie-rarchy of nodes that causes this problem. Yet it is the very nature of the computing nodes that makes the hierarchy necessary.

The basic structure of an ASOCS node was discussed earlier (see Fig. 1). The fact that these computing nodes have only two inputs is very important. The nodes are simple and computing their functions is very fast. They fit naturally into a hierarchical scheme that allows for bounded, O(log n) training and processing times. Because the nodes are limited to two inputs, each node is limited in the amount of information it can physically represent. As shown in figures 3 and 4, it may take several nodes to represent an instance or an instance set. (Refer again to figures 3 and 4. In these models each node computes the *AND* of its inputs). Representations of instances are "built" by feeding the output of some nodes into the inputs of others.

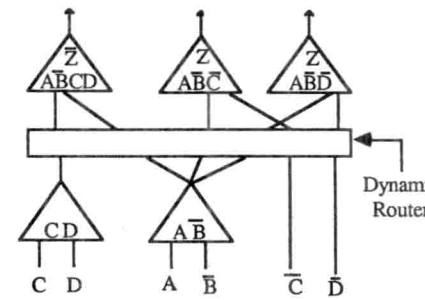

Fig. 3 - Dynamic Interconnect

Thus, it is clear that the problem with allocating unused nodes is tied to the existence of the hierarchy of nodes, which exists in turn because of the structure of the nodes. DNA's solution to the problem is to get rid of the hierarchical structure. As will be seen in the next section, this necessitates a change in the node structure.

DNA's Solution. DNA offers a solution to the problem of the unused nodes. There is no need for direct communication between nodes, so there is no hierarchical structure between nodes. This is because each DNA node in the network represents a complete instance in the *current representation* of the instance set. (For this reason DNA and similar models that are currently being developed are called Node-Per-Instance, or NPI models.) DNA nodes are capable of dynamically changing to represent new and modified instances. DNA nodes contain more information than nodes in other models. This is an important difference from previous models which, as explained above, must use a hierarchy of nodes to represent an instance. DNA nodes are logically independent of one another, so nodes can be placed anywhere in the network. Figure 5 shows the instance set of Figures 3 and 4 as it might be represented in the DNA model:

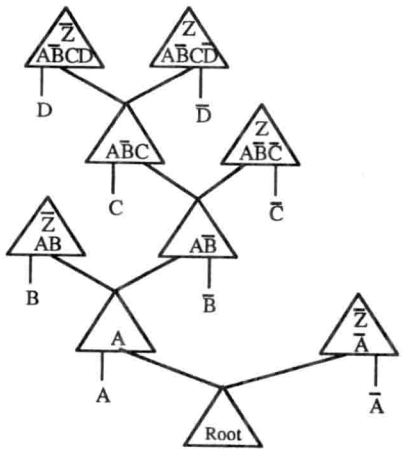

Fig. 4 - Static Interconnect

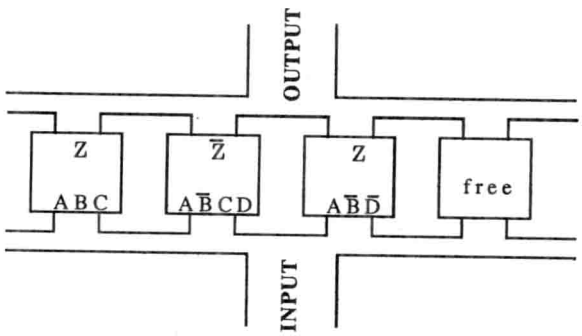

Fig. 5 - DNA nodes are independent

The bus represents the communication mechanism for broadcast of inputs and collection of outputs. ("Bus" here describes the functionality of the interconnect, but does not imply a specific implementation.) DNA has the ability to use any node in the network easily to represent instances, whereas this is not true of other models.

In order to get this flexibility and still maintain the advantages that other models have, the DNA node structure must capture what the hierarchy was doing in other models. There are a variety of ways to do this. One is to build each node as a separate hierarchical structure. Another is to time-multiplex input variables across a two-input logic gate with some extra memory at the node. A third is to use a PLA-like structure. How ever the node is actually implemented, it can logically be thought of as a control unit and an n-input logic gate, shown in figure 6 (as compared to the two-input gate of Fig.1). This difference in nodes (whether logical, physical or both) is the foundation of the differences between the DNA model and models that use node hierarchies.

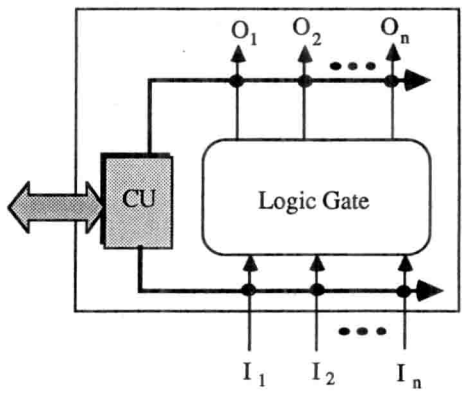

Fig. 6 - Logical Structure of DNA node

Processing and Learning in the DNA model. As in other ASOCS models, DNA has two modes--processing and learning. In processing mode, the inputs are broadcast to each of the nodes in parallel. Each node, *independent of all others*, determines whether or not it matches the inputs, and thus whether or not to put its outputs on the output bus. Since the representation of a given instance set is consistent, the outputs will always be consistent [3]. In the case that the input is matched by no node, two choices are available: either the network can output a "don't know" state, or choose the output for the instance that most closely matches the input. Which one is used is a matter of preference in implementation, but does not affect the model.

Figure 5 uses a bus, which has linear propagation time, to connect the nodes. However, because they are logically independent of one another, it does not matter how the nodes are arranged topologically as long as they can receive input and send output. Therefore a mesh, a cube, a tree, or some other polynomial or logarithmic structure can easily be used as an interconnect. It is important to note that while signals may have to pass through nodes to get to other nodes, the latter's computation is not influenced by the former.

In learning mode, the network uses the same hardware as in processing mode. However, the output must be broadcast to the nodes as well as the input (i.e. a NI is presented to the network). As above, each node independently determines how well it matches the information. A node does not put its outputs on the output bus as it might do above at this point. Instead, a node may change its function based on the result. The goal of any change is to make the overall function represented by the network consistent, so that there are no conflicting outputs. Since each node has all the information it needs, adaptation can take place asynchronously and concurrently among nodes (as in the other models).

Conclusion. The development of the DNA model is an important step from the abstract model toward the physical realization of an ASOCS machine that can be used to solve problems. It is designed to better fit the requirements for VLSI implementation than its predecessors. The advantages and limitations of the model should be seen with this in mind. Its advantages are:

1. It alleviates the unused node allocation problem that current models have. Unused nodes are easier to use in DNA.
2. The network no longer depends on a logical hierarchy of nodes for success. Physical location in the network is no longer a factor.
3. As a result of #2, instances can be placed anywhere in the network, and easily relocated without affecting the logic.
4. Different interconnect schemes may be used: dynamic, static, trees, hypercubes, busses, etc. without affecting the success of the network (although some topologies may be preferred over others).
5. As a result of #4, it is still possible to obtain fast processing and learning times as in other models, using the proper interconnect.

Its limitation is:

1. The node will be slightly more complex than nodes in current models, owing to the need for flexibility in what it represents.

Applications for ASOCS have been found in adaptive logic, system fault management, robotics and dynamic control. As an ASOCS model, and potential implementation, DNA could be used for any of these.

Current research focusses on five main areas:

1. VLSI implementation of DNA and other NPI models.
2. Use of a priority scheme with instances to limit the growth of a network.
3. New learning algorithms that improve and extend the power of current models.
4. Methods of extending functionality beyond Boolean to higher-order functions and more complex data.
5. Finding new applications for DNA.