LEGIBILITY NOTICE

A major purpose of the Technical Information Center is to provide the broadest dissemination possible of information contained in DOE's Research and Development Reports to business, industry, the academic community, and federal, state and local governments.

Although a small portion of this report is not reproducible, it is being made available to expedite the availability of information on the research discussed herein.

LA-UR -90-2755

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

LA-UR--90-2755

DE90 016505

TITLE *PREDICTION AND CONTROL OF CHAOTIC PROCESSES USING NONLINEAR ADAPTIVE NETWORKS*

AUTHOR(S) *R. D. Jones, C. W. Barnes, G. W. Flake, K. Lee, Y. C. Lee, P. S. Lewis, M. K. O'Rourke, and S. Qian*

SUBMITTED TO *Proceedings for Nonlinear and Chaotic Processes in Plasmas, Fluids, and Solids*
Edmonton, Alberta, Canada
July 23 - 27, 1990

DISCLAIMER

The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

PREDICTION AND CONTROL OF CHAOTIC PROCESSES USING NONLINEAR ADAPTIVE NETWORKS

R. D. JONES[a,c], C. W. BARNES[a], G. W. FLAKE[a,b,h], K. LEE[c], Y. C. LEE[b,f,g], P. S. LEWIS[d], M. K. O'ROURKE[a,b] and S. QIAN[b]

[a] *Applied Theoretical Physics Division*
[b] *Center for Nonlinear Studies*
[c] *Defense Research Applications / Defense Initiatives Office*
[d] *Mechanical and Electrical Engineering Division*
University of California
Los Alamos National Laboratory
Los Alamos, New Mexico 87545 USA
[e] *rdj@lanl.gov*

and

[f] *Department of Physics and Astronomy*
[g] *Institute for Advanced Computer Science*
[h] *Department of Computer Science*
University of Maryland
College Park, Maryland 20740 USA

ABSTRACT

We present the theory of nonlinear adaptive networks and discuss a few applications. In particular, we review the theory of feedforward backpropagation networks. We then present the theory of the Connectionist Normalised Linear Spline network in both its feedforward and iterated modes. Also, we briefly discuss the theory of stochastic cellular automata. We then discuss applications to chaotic time series, tidal prediction in Venice lagoon, finite differencing, sonar transient detection, control of nonlinear processes, control of a negative ion source, balancing a double inverted pendulum and design advice for free electron lasers and laser fusion targets.

1. Theory

A young child can learn to operate a complex nonlinear process, walking or riding a bicycle for instance, with no knowledge of physics or differential equations. A juggler can balance an inverted broom stick with only coarse knowledge of the state of the system. A human being seems to be able to act with reasonable precision in a complex highly interactive world with partial or partially incorrect information. One wonders, then, if the traditional deductive approach of physicists and engineers to prediction and control cannot be augmented by a more inductive point of view. Does one, for example, need to know the details of a plasma instability in order to predict and control it? Can the behavior of a chaotic process be predicted with no initial knowledge of the dynamics or dimensionality of the system? Taking life as an existence proof, we are motivated to look at some inductive and adaptive systems and their interaction with complex nonlinear phenomena. In particular, we are interested in how inductive systems can be used to predict or control the behavior of nonlinear phenomena. In this section we will review the feedforward backpropagation network, describe a new adaptive local network, and describe the computationally more powerful stochastic cellular automaton. In Sec. 2 a number of real world applications will be discussed.

1.1 Feedforward Backpropagation Network

The most common nonlinear adaptive network is the feedforward backpropagation network [1]. This net is illustrated in Fig. 1. The net is composed of input and output layers, and one or more hidden

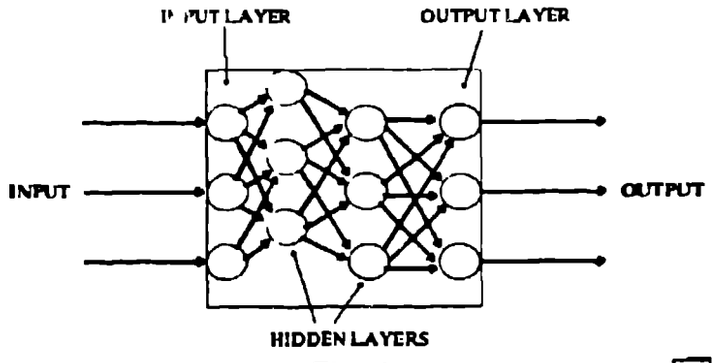

Fig. 1

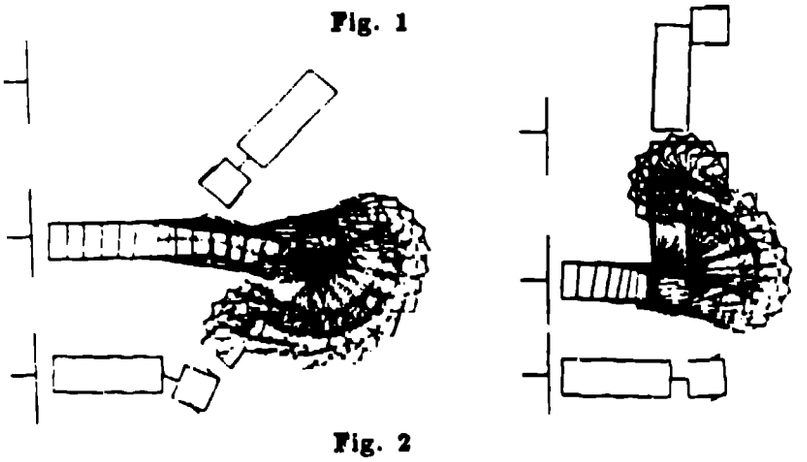

Fig. 2

layers of neurons. Information flow is in a single direction, in this case to the right. There are usually two or more layers other than input or output layers. These are called hidden layers. The output, y_i, of the i^{th} neuron is given by

$$y_i = sig(\Sigma_j W_{ij} y_j + \theta_i) \qquad 1.1-1$$

where y_j is the output of the j^{th} neuron in a layer immediately to the left of the layer in which the i^{th} neuron is located. The sigmoid function, sig, is a rounded Heaviside step function,

$$sig(x) = (1/2)[1 + tanh(x)]. \qquad 1.1-2$$

The form of this function is chosen to mimic, in a rough sense, biological neurons. The weights, W_{ij}, and thresholds, θ_i, are determined by least mean square minimization. Define a cost function,

$$E = (1/2)\Sigma_{p=1}^{M}[g(\vec{x}_p) - \phi(\vec{x}_p)]^2 \qquad 1.1-3$$

where $\vec{x}_p$ is an input training vector, $g(\vec{x}_p)$ is the training output for the input, $\vec{x}_p$, and $\phi(\vec{x}_p)$ is the network output for the training input, $\vec{x}_p$. The summation is over all training points. We do not rule out the possibility that a training point can be shown to the net more than once. In fact, that is the usual situation. Therefore, M is the number of times that any training point is shown to the net. For convenience, we have assumed a scalar output. There is no fundamental reason that there could not be multiple outputs. The learning algorithm is simply the numerical technique for the minimization of E. Common minimization methods, for instance, are gradient descent, conjugate gradient, and Newton's method.[2]

Backpropagation networks have had some impressive successes. This class of networks has been able to beat most of the traditional methods in the accuracy of time series prediction.[3,4]

Another impressive example is that of control of a backing truck (Fig. 2).[5] Two nets are used in this problem, one to learn how a truck works and the other to learn how to back a truck to a loading dock given that the net knows how a truck works. The truck is only permitted to back, not drive forward.

Backpropagation networks have some problems, however: (1) They require a great deal of training data for reasonable accuracy; (2) Interpolation is poor without a great deal of training; (3) Backpropagation nets are much slower, for comparable accuracy, than the best non-neural methods.[6] In many practical problems large amounts of data are not available. At minimum, a neural net must be able to automatically interpolate and extrapolate from a small data set. Also, in many problems, learning must occur in real time. Slow learning limits the application of neural nets.

1.2 Connectionist Normalized Linear Spline (CNLS) Network: Feedforward Operation

Learning speed has been addressed by Moody and Darken.[7] They replaced the backpropagation net of sigmoidal nonlinear elements with a net of local radial basis functions. Much of the reduction in learning time is due to the fact that there is only one hidden layer in a radial basis net while there are typically two or more in a backpropagation network. Moody and Darken estimate the reduction in learning rate on the Mackey-Glass equation for comparable accuracy to be a factor of 10^2-10^3. Casdagli[8] has shown how this net may be used to obtain invariants of a chaotic time series.

Radial basis function nets do not, however, address the problem of excessive training data or poor interpolation. To achieve comparable accuracy to the backpropagation benchmark of the Mackey-Glass equation, several more neurons are required. More significant, the number of training data points required to train the net is greater by a factor of 27.

A natural evolution is to modify radial basis function nets in a manner that improves interpolation and reduces the amount of training necessary for accurate learning.[9-12] Note the identity,

$$g(\vec{x}) = \frac{\Sigma_{j=1}^{N} g(\vec{x})\rho_j(\vec{x})}{\Sigma_j \rho_j(\vec{x})}. \qquad 1.2-1$$

Here, $\rho_j(\vec{x})$ is a localized function of $\vec{x}$ about some $\vec{a}_j$. Hence, $g(\vec{x})$ on the right of Eq. 1.2-1 can be approximated by its Taylor expansion about $\vec{a}_j$. We have then,

$$\phi(\vec{x}) = \Sigma_{j=1}^{N}[f_j + (\vec{x} - \vec{a}_j)\cdot\vec{d}_j]\frac{\rho_j(\vec{x})}{\Sigma_j \rho_j(\vec{x})} \qquad 1.2-2$$

for an approximation to $g(\vec{x})$. This net differs from the radial basis function net of Moody and Darken[7] in two ways, the use of normalization and also a linear term, $(\vec{x}-\vec{a}_j)\cdot\vec{d}_j$. The use of a normalization term was suggested but not pursued by Moody and Darken. The addition of these two terms is responsible for the reduction in the amount of training data needed to obtain reasonable approximations. As in the case with radial basis functions, the training of f_j and $\vec{d}_j$ is linear and hence very fast. The widths of the basis functions can also be trained. This training is nonlinear. We will show, however, for the learning algorithm we adopt that this training can also be very fast. In some applications the components of the input training vector are of different types. For example, one might want to include both slope and curvature information in the input vector $\vec{x}$. A radial function of $(\vec{x} - \vec{a}_j)^2$, which weights every component equally might not be appropriate. In these situations differential weighting of the components can be employed. The basis function will then be wider in some directions than others. Care must be used when training the widths differentially as multi humped basis functions can result. The locations of the basis function centers can also be trained. We typically use a genetic algorithm for this training.

Learning algorithms can be either on-line or off-line. Off-line algorithms attempt to calculate weights without any reference to time ordering of the training data. Thus, all the training data must be collected before training can start. On line algorithms, on the other hand, attempt to modify weights as information in the form of training data flows in. On line algorithms are able to handle varying amounts of training data and are able to modify the system in the presence of drift in the conditions. This is very difficult with an off line algorithm. Additionally, on line algorithms are able to handle

amounts of input data that would severely tax memory storage capacities if an off-line method were used. Most neural nets are trained, therefore, with on-line methods.

On-line methods themselves come in two extremes. The method can remember all the data that it has been shown up to the present or it can only be aware of the training set it is being shown at the present. Conjugate gradient learning and multidimensional Newton's method fall into the former category. We will use a method that falls into the latter category. Since we will use less information, we will pay a price in accuracy, but this is compensated by speed and simplicity.

The most important weights to be trained in the CNLS net are the linear weights f_j and $\vec{d}_j$. The method is essentially an iterative Newton's method and is described in detail in reference 10. The recursion relations are

$$f_j^{p+1} = f_j^p + \frac{1}{3}[g(\vec{x}_p) - \phi(\vec{x}_p)]\frac{\rho_j(\vec{x}_p)\Sigma_k \rho_k(\vec{x}_p)}{\Sigma_k \rho_k^2(\vec{x}_p)} \qquad 1.2-3$$

and

$$\vec{d}_j^{\,p+1} = \vec{d}_j^{\,p} + \frac{1}{3}[g(\vec{x}_p) - \phi(\vec{x}_p)]\frac{(\vec{x}_p - \vec{a}_j)\rho_j(\vec{x}_p)\Sigma_k \rho_k(\vec{x}_p)}{\Sigma_k(\vec{x} - \vec{a}_k)^2 \rho_k^2(\vec{x}_p)}. \qquad 1.2-4$$

where the subscripts (superscripts) p indicate the p^{th} presentation of a training pattern.

Following Lapedes and Farber[3] and Moody and Darken[7] we test the network on the evolution of the logistic map. Here, the network must approximate the function

$$x_{n+1} = 4x_n(1 - x_n).$$

The input to the net is x_n and the output is x_{n+1}. The results are displayed in Fig. 3. As in the case of Lapedes and Farber and Moody and Darken, we use five neurons (basis functions) in our hidden layer. These authors used, in addition, a linear connection between the input neuron and the output neuron. There are a total of 10 weights total to be trained in our net. This is to be compared with 17 weights in the backpropagation net of Lapedes and Farber and 15 for Moody and Darken's net. The previous authors used 1000 input-output pairs to train their nets. Initially, we use four. The locations of the four training points are marked with diamonds and arrows in Fig. 3. The centers of the basis functions are marked with long dashes on the abscissa of the top plot in Fig. 3. Conjugate gradient training was used in the backpropagation and radial basis function nets, while we have used the one dimensional Newton's method of Eqs. II-21 and II-22. As an initial guess for the weights we chose $f_j = g(\vec{a}_j)$ and $\vec{d}_j = 0$. The functional form for the basis functions were taken to be Gaussian, $exp(-\beta(\vec{x} - \vec{a}_j)^2)$, where $\beta = 1/0.5^2$. The approximation based on this initial guess is displayed. The four training points were shown to the net randomly and in this training run the net saw a training point 4000 times (1000 times, on average, per training pair).

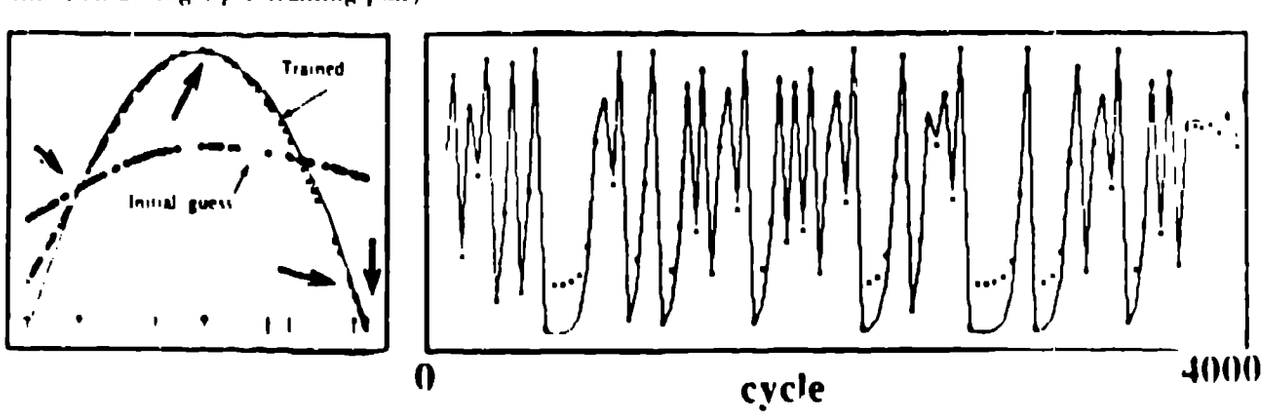

FIG. 3

The net performs well even when presented with only four training points. It can be seen that a smooth and reasonable interpolation obtains. The worst fit occurs in the lower left portion of the parabola where the net is forced to extrapolate. The error here is 5 to 10 percent. Prediction of a novel time series is displayed in the middle plot. It can be seen from the absolute error curve on the training points that most of the learning occurred in the first 20 training cycles. Slow improvement is seen after that.

If the net is trained on 100 points, the prediction accuracy improves Once again the learning occurs on a fast and a slow time scale.

Training on the widths, β_j, improves accuracy even more. The total prediction error drops to less than a percent and becomes comparable to the error of approximately 0.5 percent using backpropagation or radial basis functions with conjugate gradient learning (Moody and Darken 1989). With training on widths, the total number of adjustable weights in the net increases to 15 which is comparable to the backpropagation and radial basis function tests. The total number of training points, here, is 100 compared with 1000 used by the other nets. Training took a few minutes of SUN 3/75 time as compared with an hour of SUN 3/50 time for the other two nets.

1.3 CNLS Network: Recurrent Operation

The CNLS net works reasonably well for smooth function approximation. There are many instances, however, when one encounters discontinuities and singularities and a smooth approximation is inadequate. A discontinuity very familiar to physicists is the shock wave. A simple description of a shock wave is given by[13]

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} = 0 \qquad 1.3-1$$

where v is the fluid velocity and t and x are temporal and spatial coordinates, respectively. Eq. 1.3-1 has the recurrent solution

$$v = v(x - vt). \qquad 1.3-2$$

Faster moving fluid elements can overtake slower moving elements with the net result that an initial shallow velocity profile is steepened. Eq. 1.3-2 can be solved by iterating on an initial solution from $t = 0$ to some final time, t. This is reminiscent of the generation of fractals by iterated function systems.[14]

Motivated by the shock wave and fractal examples we examine the possibility of treating discontinuities by iterated adaptive networks.[15] To incorporate this recurrency into the CNLS net one extra input neuron is added for each output neuron (Fig. 4). These additional 'feedback neurons' are used to feed the previous state of the network to itself. The variable t is a scalar which rescales the input.

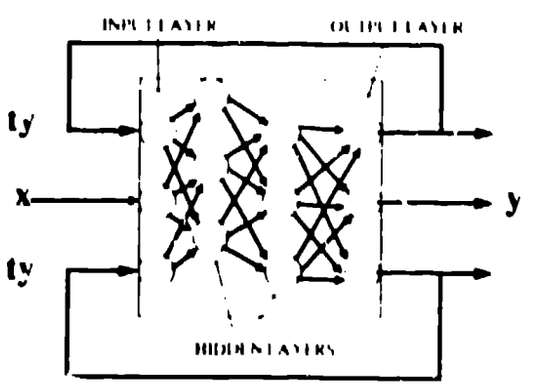

FIG. 4

In the function evaluation phase, the network is iterated many times (5-20) for each input vector. The functions converge to fixed points if the net has been properly trained. The initial guess for the iteration is obtained from the neural net in feedforward mode.

If t is chosen too high, the network will rely too heavily upon the feedback neurons, resulting in poor performance independent of the input vector. If t is too low, however, performance will approach the smooth approximation of the feedforward net.

The fixed points are determined by training in a feedforward mode. The desired outputs are used as inputs to the feedback neurons as well as for error correction at the output of the net. The network tends to generate surfaces through the fixed points with shallow derivatives ensuring stability of the fixed points.

The ability of the net to identify and approximate discontinuities is displayed in Fig. 5.

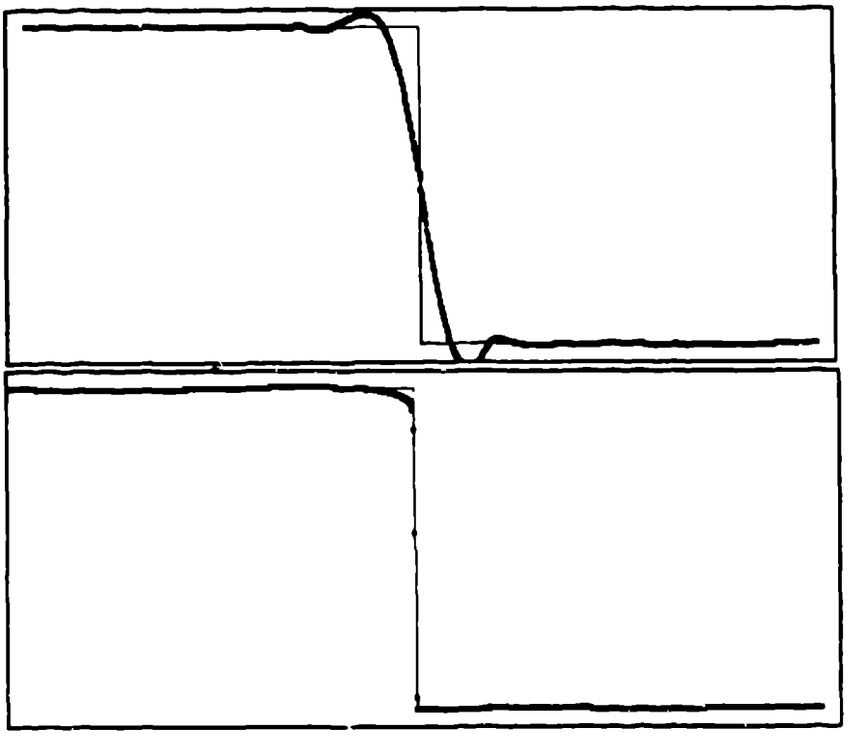

FIG. 5 The upper plot illustrates the approximation of a step function by the feedforward CNLS network. The dark line is the approximation. Note the "Gibbs' " phenomena around the discontinuity. In the lower plot the CNLS net in recurrent operation was used to approximate the step function.

1.4 Adaptive Stochastic Cellular Automata

An adaptive network in recurrent operation is an example of a finite state machine.[16] Finite state machines are able to recognize regular grammars. In actual fact, however, adaptive networks have only demonstrated regular grammatical inference of "toy" problems. More sophisticated inference can be performed if memory is added to finite state machines. We can add memory to adaptive networks by linking together recurrent networks so that the output depends not only on the current output and input of the network but also on the output of neighboring networks. A network of finite state machines linked in this manner is called a cellular automata.[17] It has been conjectured that even simple cellular automata are able to perform universal computation.[18] In our cellular automata the finite state

machines are adaptive. The space of weights can be of very high dimension. In order to more efficiently explore such a large space we use genetic techniques. The input-output mappings are thus stochastic.[19] The inputs, outputs, and weights are usually simply encoded into binary form.

2. Applications

In this section we briefly describe a few of the applications of the CNLS net in both feedforward and recurrent operation and of the Adaptive Stochastic Cellular Automaton.

The city of Venice, Italy has sunk approximately 25 *cm* in the last 100 years. Flooding has become a serious problem. Because of pollution in Venice Lagoon, attempts to control the flooding with gates must permit unimpeded flow whenever high water is not expected. It is therefore important to be able to predict when flooding will occur with some accuracy. There is a regular astronomical component to the tides in Venice Lagoon as well a possibly chaotic component due to meteorological phenomena. We have used the CNLS net in feedforward mode to predict tidal behavior. The results are illustrated in Fig. 6.

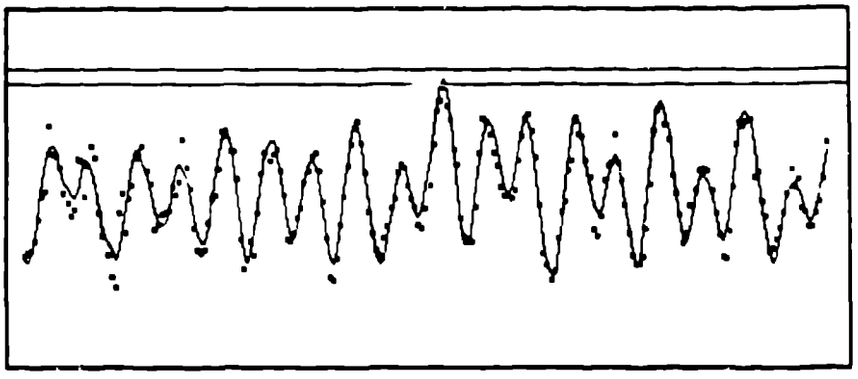

FIG. 6 Ten days are displayed. Prediction was three hours into the future based on the tidal data for twelve hours into the past. Twelve basis functions were used. Training was on 1984 data. Prediction is on 1985 data. The two horizontal lines correspond to the 100 *cm* and 110 *cm* marks. A siren is sounded when a local committee in Venice predicts the tide will exceed 1[illegible] *m*.

We can use the adaptive stochastic cellular automata to balance an inverted ***double*** pendulum.[20] The double pendulum system that we want to control with the stochastic CA is a cart-pole system.[21] The system is controlled by applying a force to a cart on which the two poles are to be balanced. In addition to one pole hinged on the cart, we hinge another pole on the top of the first pole. Balancing a single pole is a problem with one unstable degree of freedom (angle θ). Balancing an inverted double pendulum is a problem with two unstable degrees of freedom (angle θ_1 and θ_2). If the double pole system is to be controlled by a force f_r on the cart the two poles must oscillate on sufficiently different time scales. To make the system easy to control, the first pole is chosen to be 10 times longer than the second pole. The characteristic time scale is about three to one.

The parameters used in simulation are: cart weight is 1 kg. The first pole weighs 0.1 kg and is 10 meters long, and the second pole weighs 0.1 kg and is one meter long. In this example, we only control θ_1 and θ_2.

The initial condition is chosen randomly with $-3° < \theta_1 < 3°$, $-5° < \theta_2 < 5°$, and some small angular velocities. The time step in reinforcement learning is 0.015 sec. The input variables are coded into 64-bit strings.

The stochastic CA is rewarded or penalized according to an adaptive critic, and when either θ_1 or θ_2 reaches $\pm 10°$, a fraction of the previous actions are punished. After about two thousand failures,

the adaptive controller is able to balance the inverted double pendulum system for two hours simulated real time, which is the time limit in the simulation.

Other applications which we have considered[12] but which we do not address here are application of pattern recognition techniques and a simplified CNLS net to detection of transients in sonar signals, using adaptive networks to speed up finite differencing, control of beam quality in a negative ion source,[11] and design advisors for Free Electron Lasers and laser fusion targets.

3. Concluding Remarks

Nonlinear adaptive networks are still in their infancy. As with any new field there is optimism to the point that often the capabilities of the networks are oversold. On the other hand, the field has advanced to the point that it is clear that there will be a computational niche that can be filled by adaptive computing. It is likely that networks of the future will look little like the feedforward backpropagation networks in common use today. Already we see how to reduce the data requirements and increase the learning speed by a few orders of magnitude through a more careful approach to architectures and learning algorithms. It is also becoming clear how to increase the computational ability of the networks through the use of networks of networks.

4. References

1. P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences." ***Masters thesis, Harvard University.*** (1974); D. B. Parker, "Learning logic." ***Invention Report S81-64, File 1, Office of Technology Licensing, Stanford University, Stanford, CA.*** (1982); D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning internal representations by error propagation. in ***Parallel Distributed Processing***, eds. D. E. Rumelhart and J. L. McClelland, **1** (1986) 318-362.

2. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, ***Numerical Recipes,*** (Cambridge University Press. Cambridge, 1986)

3. A. S. Lapedes and R. Farber, "Nonlinear signal processing using neural networks: Prediction and system modeling." ***Technical Report, Los Alamos National Laboratory, Los Alamos, New Mexico 87545*** (1987).

4. A. S. Lapedes and R. Farber, " How neural nets work." in ***Neural Information Processing Systems,*** ed. D. Z. Anderson, (AIP, New York, 1988) 442-456.

5. D. Nguyen, and B. Widrow. "The truck backer-upper: An example of self-learning in neural networks." ***Proceedings IJCNN, Washington (June 1989),*** (1989) pp. II-357-363.

6. J. D. Farmer and J. J. Sidorowich, "Predicting chaotic time series," ***Physical Review Letters,*** **59**, (1987) 845.

7. J. Moody and C. J. Darken, "Fast learning in networks of locally tuned processing units," ***Neural Computation,*** **1**, (1989) 281-294.

8. M. Casdagli, "Nonlinear prediction of chaotic time series," ***Physica D,*** **35**, (1989) 335-356.

9. Y. C. Lee, "Neural networks with memory for intelligent computations," in ***Proceedings of the 13th Conference on the Numerical Simulation of Plasmas. Santa Fe, New Mexico, September 17-20, 1989.*** (1989).

10. R. D. Jones, Y. C. Lee, C. W. Barnes, G. W. Flake, K. Lee, P. S. Lewis and S. Qian, "Function approximation and time series prediction with neural networks," ***Talk presented at the Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, New Mexico on September 27, 1989.***

Los Alamos Report LA-UR-90-21 and in *Proceedings of the International Joint Conference on Neural Networks, San Diego, California, June 17-21, 1990.* (1990) I-649-666.

11. J. A. Howell, C. W. Barnes, S. K. Brown, G. W. Flake, R. D. Jones, Y. C. Lee, S. Qian, and R. M. Wright, "Control of a negative-ion accelerator source using neural networks," *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems Vancouver, B.C., Canada October 30-November 3, 1989. Los Alamos report LA-UR-89-3597.* (1989).

12. R. D. Jones, "Nonlinear Adaptive Networks: A Little Theory, a Few Applications," *Proceedings of the First Los Alamos Workshop on Cognitive Modeling in System Control: Theoretical Foundations and Prospects for Applications, June 10-14, 1990, Santa Fe, New Mexico.* (1990).

13. R. C. Davidson, *Methods of Nonlinear Plasma Theory*, (Academic Press, New York, 1972), pp. 15-18.

14. M. Barnsley, *Fractals Everywhere*, (Academic, San Diego, 1988).

15. M. K. O'Rourke, R. D.Jones, Y. C. Lee, C. W. Barnes, G. W. Flake, K. Lee and S. Qian, "Recurrency in feedforward networks for prediction of high dimensional cliffs," *Los Alamos Report (in preparation).* (1990).

16. M. D. Davis and E. J. Weyuker, *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science,* (Academic, San Diego, 1983).

17. S. Wolfram, *Theory and Applications of Cellular Automata,* (World Scientific, Singapore, 1986).

18. A. R. Smith, "Simple computational-universal cellular spaces and self-reproduction," in *Conference Record, IEEE 9th Annual Symposium on switching and automata theory,* (1968) 269-277; J. W. Thatcher, "Universality in the von Neumann cellular model," in *Essays on cellular automata,* ed. A. W. Burks, (University of Illinois Press, 1970) pp. 132-186; E. F. Codd, "Propagation, computation and construction in two-dimensional cellular spaces," in *Cellular Automata,* (Academic, New York, 1965); J. von Neumann, "The general and logical theory of automata," in *Cerebral Mechanisms in Behavior: The Hixon Symposium,* (John Wiley and Sons, 1951).

19. Y. C. Lee, S. Qian, R. D. Jones, C. W. Barnes, G. W. Flake, M. K. O'Rourke, K. Lee, H. H. Chen, G. Z. Sun, Y. Q. Zhang, D. Chen, and G. L. Giles, "Adaptive stochastic cellular automata: theory," *Physica D* to be published.

20. S. Qian, Y. C. Lee, R. D. Jones, C. W. Barnes, G. W. Flake, M. K. O'Rourke, K. Lee, H. H. Chen, G. Z. Sun, Y. Q. Zhang, D. Chen, and G. L. Giles, "Adaptive stochastic cellular automata: applications," *Physica D* to be published.

21. A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics SMC-13* (1983) pp. 834-846.