Modular Learning in Neural Networks

Dana H. Ballard
Department of Computer Science
University of Rochester, Rochester, New York 14627

Abstract

In the development of large-scale knowledge networks, much recent progress has been inspired by connections to neurobiology. An important component of any "neural" network is an accompanying learning algorithm. Such an algorithm, to be biologically plausible, must work for very large numbers of units. Studies of large-scale systems have so far been restricted to systems without internal units (units with no direct connections to the input or output). Internal units are crucial to such systems as they are the means by which a system can encode high-order regularities (or invariants) that are implicit in its inputs and outputs. Computer simulations of learning using internal units have been restricted to small-scale systems. This paper describes a way of coupling autoassociative learning modules into hierarchies that should greatly improve the performance of learning algorithms in large-scale systems. The idea has been tested experimentally with positive results.

1. Introduction

An important component of any artificial intelligence system ultimately will be its ability to learn. Very recently there has been great progress in the development of learning algorithms [Rumelhart et al., 1986 (1); Rumelhart and Zipser, 1985 (2); Ackley et al., 1985 (3); Pearlmutter and Hinton, 1986 (4); Lapedes and Farber, 1986 (5)].

All of the above algorithms use internal representations to represent regularities in the environment. The internal representations capture efficient encodings of the environment that presumably facilitate the behavioral needs of the system. These individual algorithms have their own advantages and disadvantages, but a common question related to all of them is *whether or not they scale with the size of the problem.* In other words, even on an appropriate parallel architecture, the computational complexity in the average case may not remain constant or at worst scale with the problem size. The result is that it is likely that additional insights will be needed to implement learning algorithms in massively parallel systems.

2. Hierarchies

The tremendous advantage of hierarchies as a compact encoding of input-output pairs is the principal motivation for developing a learning algorithm that is geared to developing hierarchical encodings. One possibility is to use the Backpropagation algorithm with several internal levels. Our computer experiments in Section 4, however, show that this formulation does not seem to have good scaling properties. An example that took 256 iterations to converge with one internal layer took over 4096 iterations to converge with three internal layers. Thus we were motivated to develop a modular reformulation of Backpropagation learning with better convergence properties.

Another idea that we will use is that of autoassociation. Consider first a simple modification of the Backpropagation algorithm that is shown in Figure 1. The figure shows the standard three-layer architecture used in most experiments. We will refer to the layers as the input, internal, and output layers, as shown on the figure. The number of units at each layer we term the *width* of the layer.

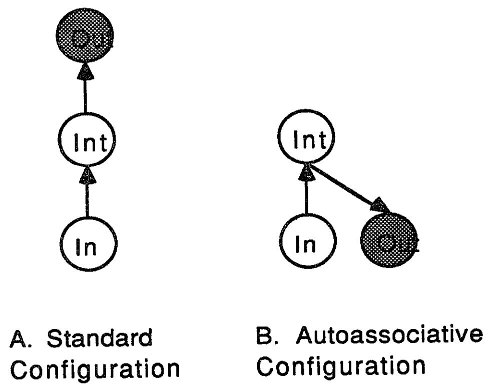

Figure 1. In the autoassociative configuration the output is constrained to be identical to the input.

To simplify what follows, we neglect the width of the layers and just use a representative unit for all the units in a layer. Let us use two-way connections so that now the internal units are connected to the *input* units. Note that now the same learning algorithm can be adapted to this special problem of predicting the input. Activation from the input is propagated to the internal units and then back to the output, where it can be interpreted as a virtual copy. That is, it is the input as reconstructed from the internal representation. Since the input is known, this can be used to generate an error signal, just as in the feedforward case, that is then sent backwards around the network to the input weights. This architecture was proposed by Hinton and

Rumelhart and has been recently studied by Zipser [1986].

3. Learning with Modular Hierarchies

The main result of this paper is to show how a purely autoassociative system system can be modularized in a way that is resistant to changes in problem scale. Consider Figure 2, which describes the general idea. Consider that an autoassociative module is used to learn a visual representation. Now imagine that a similar process takes place at the output (motor) level, where in this case the system is codifying efficient internal representations of quickly varying movement commands. Both the motor and visual internal representations, being codes for the more peripheral representations, will vary less. This means that if one views the situation recursively, at the next level inward the problem of encoding peripheral representations is repeated, but now it is cast in terms of *more abstract, more invariant* representations of the peripheral signals. It also means that the same principles can be applied recursively to generate a set of learning equations that are geared to the new levels. Thus one can imagine that the abstract visual and motor levels are coupled through another autoassociative network that has a similar architecture to the lower levels but works on the abstractions created by them rather than the raw input. The next autoassociative module, termed ABSTRACT in Figure 2, starts with copies of the internal representations of the SENSORY and MOTOR modules and learns a more abstract representation by autoassociation.

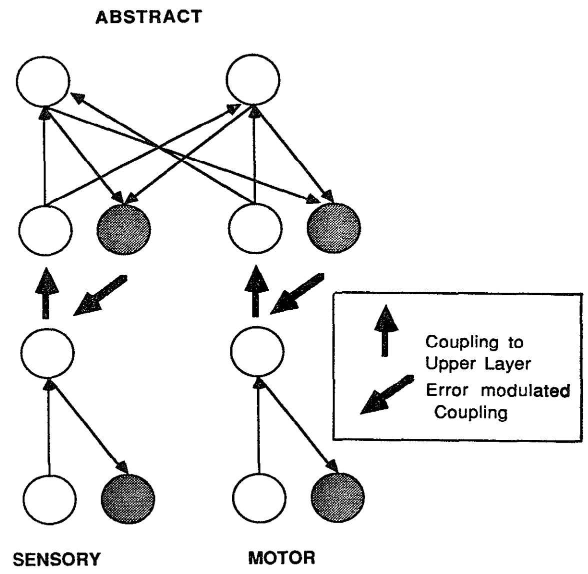

Figure 2. The main idea : Peripheral modules can work in an almost decoupled fashion to build more abstract representations. These are tightly coupled by more abstract modules that build still more abstract representations. The depth of two in the figure is purely schematic: the principle extends to arbitrary levels.

Perhaps the biggest advantage that has occurred with this reformulation is that the equations at each level can be thought of as being relatively decoupled. This means that they can be run in parallel so that the error propagation distances are short. In practice one would not want them to be completely decoupled, as then higher-level representations could not effect lower-level representations. At the same time, problems may occur if the different levels are coupled too tightly before the system has learned its patterns, since in this case errorful states are propagated through the network. Here again the hierarchical reformulation has a ready answer: since there is now a measure of error for each layer, the activation of the upper, output levels can be coupled to the lower levels by a term that tends to zero when the error is large and one when the error is small.

To develop the solution method in more detail, consider the error propagation equations from [Rumelhart et al., 1986]. They minimize an error measure $E = \Sigma_p E_p$, where

$$E_p = \sum_i (s^d_{ip} - s_{ip})^2$$

where the subscript i ranges over the output units and the d denotes the desired output. The unsubscripted s denotes the actual output. In what follows, the subscript p, which ranges over individual patterns, will be dropped for convenience. For a two-layered system such as is characterized in Figure 1A, the equations that determine the activation are given by:

$$s_j = \sigma(\sum (w_{ji} s_i + \theta_j)$$

where $\sigma(x) = 1/(1 + e^{-\beta x})$. The output of the jth unit s_j ranges between zero and one. The synaptic weights w_{ji} and threshold θ_j are positive and negative real numbers.

The equations that change the weights to minimize this error criterion are:

$$\Delta w_{ji} = \eta \delta_j s_i$$

for output units,

$$\delta_j = (s^d_j - s_j) s_j (1 - s_j)$$

and for hidden units,

$$\delta_j = s_j (1 - s_j) \sum_k \delta_k w_{kj}$$

These equations are derived in [Rumelhart et al., 1986].

Now let us consider the architecture of Figure 2. In this architecture, the connections from the hidden units feed back to the input units, so that now the prime notation has a special meaning. *It is the activation level of the input units that is predicted by the hidden units.* This is subtracted from the actual input level, which may be regarded as clamped, in order to determine the error component used in correcting the weights. Thus essentially the same equations can be used in an autoassociative mode. The elegance of this formulation is that it can be extended to arbitrary modules. Where the subscript m denotes the different modules, the equations that determine the activation are now given by:

$$s_{j,m} = \sigma(\sum w_{ji,m} s_{i,m} + \theta_{j,m})$$

The equations that change the weights to minimize this error criterion are:

$$\Delta w_{ji,m} = \eta_m \delta_{j,m} s_{i,m}$$

for output units,

$$\delta_{j,m} = (s^d_{j,m} - s_{j,m}) s_{j,m} (1 - s_{j,m})$$

and for hidden units,

$$\delta_{j,m} = s_{j,m} (1 - s_{j,m}) \sum_k \delta_{k,m} w_{kj,m}$$

Now for the coupling between modules. A module m_2 is said to be *hierarchically coupled* to a module m_1 if the activation of the input layer of m_2 is influenced by the internal layer of m_1, and also the activation of the output layer of m_2 influences the internal layer of m_1. In this case m_2 is said to be the *more abstract* of the two modules and m_1 the *less abstract.*

The modules are directly *input coupled* if the activation of a subset of the units in the input layer of m_2 is a direct copy of the activation of units in the internal layer and *output coupled* if the activation of the units in the internal level of m_1 uses the activation of the output units in its sigmoid function.

The hierarchical algorithm works as follows. Consider first the "sensory" module in Figure 2. This can be thought of as a standard autoassociative Backpropagation network. The "motor" module can be thought of in the same way. Each of these modules builds an abstract representation of its visual input in its own internal layer. Next the activation of these internal layers is copied into the input layer of the "abstract" module. In the architecture we tested, the abstract module has double the width of the sensory module, and the widths of the sensory and motor modules are equal. The abstract module learns to reproduce this input by autoassociation in its output layer. This module does two things: first, it builds an even more abstract representation of the combined visual and motor inputs; and second, it couples these two inputs so that ultimately the visual inputs will produce the correct motor patterns.

While the coupling in the upward direction is straightforward, the coupling in the downward direction is more subtle, so we will develop the rationale for it in detail. Remember that the equation for updating the weights has the following simple form:

$$\Delta w_{ji} = \eta \delta_j s_i$$

In this equation η is a parameter that must be chosen by experiment. Normally, η is constant throughout, or at least for each layer, but is this the right thing to do? Recently Baum et al. [1986] have shown that sparse encodings, where only a small fraction of the internal units are on simultaneously, have special virtues in terms of retrieval properties that are noise resistant, and Scalettar and Zee [1986] have demonstrated that sparse encodings emerge under certain experimental conditions where noise is added to the input. In addition, a straightforward argument shows that, to the extent that the internal representation can be made sparse, the learning process will be speeded up. The reason for this is that the weight change for a given pattern may not be in the same direction as that of the other patterns, so that the different weight changes may interfere with each other. Sparse encodings tend not to have this problem, since the activation of the unit whose weight is to be changed is likely to be non-zero for only a few patterns. One way to make the encodings sparse is to incorporate additional procedures into the basic learning algorithm that favor sparse representations. Scalettar and Zee [1986] used a selective weight decay where the largest weights (in absolute value) decayed the slowest. For reasons that will become clear in a moment, we changed the weight modification formula to:

$$\Delta w_{ji} = \eta \delta_j s_i (s_j / s_{j\max})$$

Under this "winner gets more" (WGM) heuristic, the unit in a layer with the most activation has its weights changed the most. In other words, the weight change was scaled by the relative activation of the unit. This heuristic had a marked positive effect on convergence. Figure 3 shows a comparison between the two formulae in the simple case of learning identity patterns using a 4-4-4 network.

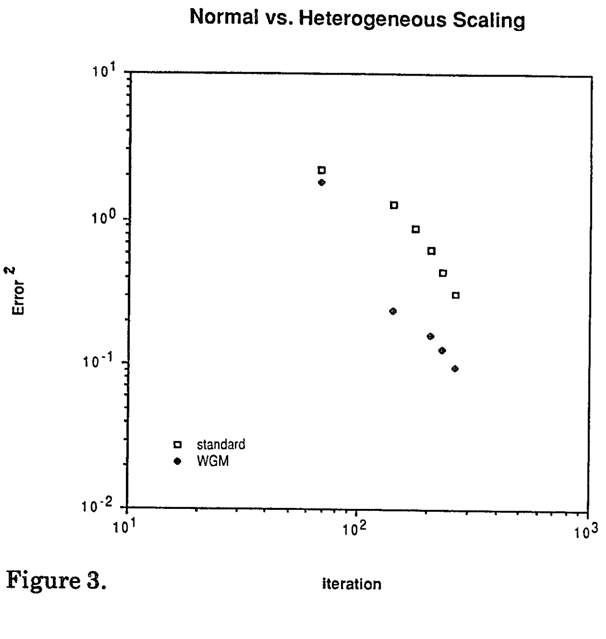

Figure 3.

This result is important because, in the limit, the downward coupling between modules will have the same effect. The argument is as follows: in downward coupling the activation from the output layer of the more abstract module is added in to that of the internal layer of the less abstract module to which it is coupled. Since the abstract module is autoassociating, its pattern should, in the limit, be identical to that of the lower module's internal layer. Thus adding this activation in is equivalent to scaling the weight change formula relative to the rest. Thus this procedure should improve convergence since it is a type of WGM strategy.

The coupling between modules is handled as follows. Suppose that the bits of the internal representation of the "sensory" module m_1 map onto the first N bits of the "abstract" module, m_3. Further, suppose the N bits of the "motor" module m_2 map onto the second N bits of the "abstract" module. Then the upward coupling is determined by:

$$s^{\text{input}}_{i,m_3} = s^{\text{internal}}_{i,m_1} \qquad i = 1, \ldots, N$$

$$s^{\text{input}}_{i+N,m_3} = s^{\text{internal}}_{i,m_2}$$

The downward coupling is more subtle. Consider the computation of the activation of the internal units of modules m_1 and m_2. The decoupled contribution is given by

$$\sum w_{ji,m} s_{i,m} + \theta_{j,m}, \qquad m = m_1, m_2$$

To this is added a term

$$\gamma (\sum w_{ij,m_3} s^{\text{internal}}_{i,m_3} + \theta_{j,m_3}$$

where γ is a function of the error in the abstract layer E_A. Where E_A is given by

$$E_A = \sum_{i=1}^{N} |\delta_{i,m_3}|$$

γ is defined by

$$\gamma = \frac{1}{1 + \alpha E_A}$$

for some $\alpha > 0$ so that $\gamma = 1$ where $E_A = 0$ and is small when E_A is large.

4. Experimental Results

The simulations all used patterns with one unit (bit) set to one (on). Thus for a four-bit input layer the patterns were: (1 0 0 0), (0 1 0 0), (0 0 1 0), (0 0 0 1), and these were paired with corresponding output patterns. This problem has been termed *the encoder problem* by Ackley et al. [1985]. The main difference in our architecture is that there are sufficient internal units so that no elaborate encoding of the pattern is forced.

As a baseline, the encoder problem was tested on 4-4-4 feedforward architecture: four input units connected to four internal units connected to four output units. The Backpropagation algorithm was used. The results of this simulation are shown below. Figure 4 shows how the squared error varies with the number of iterations. Following the simulation used by Scalettar and Zee [1986], a η of 0.75 and a β of 2.0 were used. The weights and thresholds were initialized to random numbers chosen from the interval (-0.5, 0.5), and there was no "momentum" term [Rumelhart et al., 1986].

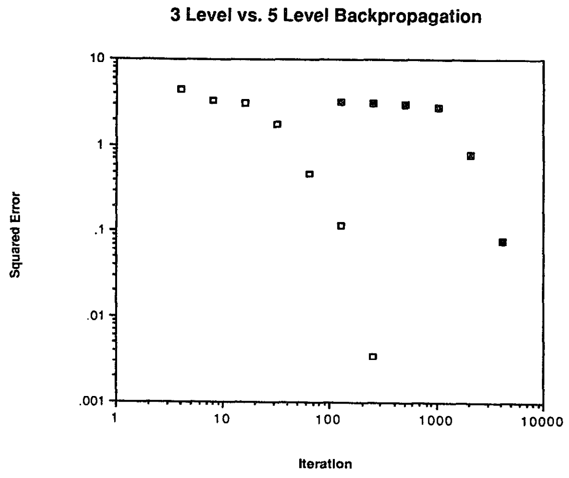

Figure 4.

Next the encoder problem was tested on a 4-4-4-4-4 architecture under the same conditions. In all the simulations tried, the algorithm found the desired solution, but took very much longer. Figure 4 compares the rate of convergence of the three-layer and five-layer networks. Although it eventually converges to the correct solution, it takes 20 times longer.

Now consider the hierarchical modular architecture. The particular hierarchical architecture that we tested can be thought of as three distinct modules. There is a 4-4-4 system that learns representations on an *input* pattern (the encoder problem), a 4-4-4 system that learns to encode the *output* pattern (in this case the encoder pattern again), and an 8-8-8 system that encodes the internal representations produced at the internal layers of both the input and output modules. The eight-bit wide system of units serves to couple the input to the output. The input and output can be thought of as at the same level in terms of the abstract hierarchy, whereas the eight-bit system is above them.

In the simulation, the state of the internal units for the input and output modules is copied into the "input" units of the more abstract module. This pattern is then learned by autoassociation using the Backpropagation algorithm. At each step the activation of the internal units of the upper module is added to that of the internal units of the lower modules, but weighted by a coupling factor. The coupling factor depends on the effort of the upper module. We used a factor γ, where

$$\gamma = 1.0 / (1.0 + 1000.0\, E_A)$$

where E_A was the average absolute error over all the patterns.

Figures 5 shows the initial and final states of the system. Figure 6 shows the error behavior, comparing the modular system to the original three-level 4-4-4 configuration.

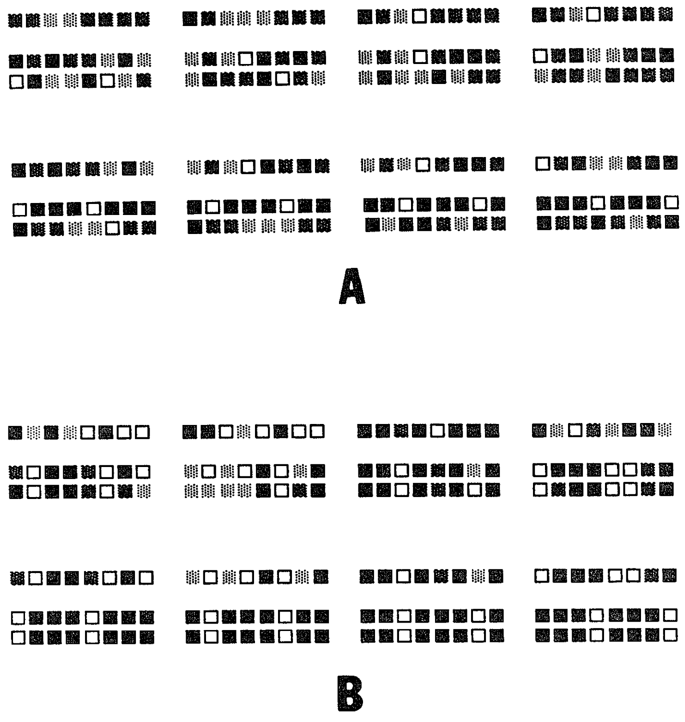

Figure 5. (A) Initial states, and (B) final states of the modular system. The four columns of units denote the response to individual patterns. The individual rows are in groups of three denoting (from top to bottom) internal, input and output units. The sensory and motor modules are 4-4-4; the abstract module is 8-8-8.

□ > 0.75 ▥ > 0.5 ▩ > 0.25 ■ < 0.25

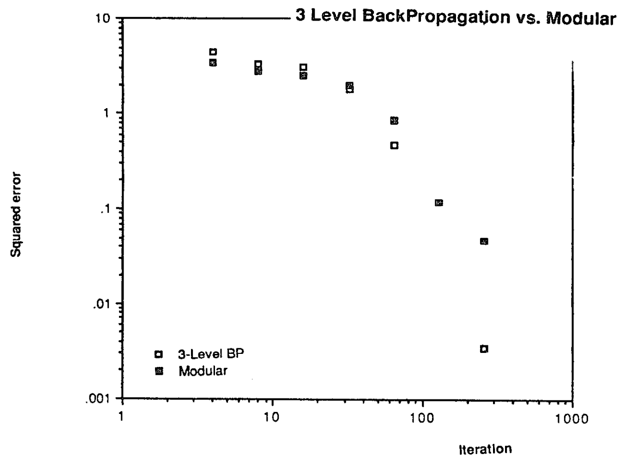

Figure 6.

These results are very positive, as they show that the convergence of the modular system is comparable to that of the original 4-4-4 system. Experiments are now underway on larger systems to try and confirm this initial result.

5. Discussion and Conclusion

The theme of this paper is that a variety of technical problems make the credit assignment problem difficult for extensively connected, large-scale systems. Realistic models will eventually have to face these problems. The solutions offered here can be summarized as attempts to break the symmetry of a fully connected layered architecture. The module concept breaks symmetry by having selective rules for error propagation. The weights for different units have different modification rules.

The main technical result of this paper is to show that multiple-layer hierarchical systems can be built without necessarily paying a penalty in terms of the convergence time. To do this it was shown that the Backpropagation algorithm, when cast in terms of an autoassociative, three-layer network, could be made into a modular system where modules at different levels of abstraction could be coupled. Almost all of the results are based on empirical tests, although there are some analytical arguments to suggest that the experimental results should have been expected. The experimental results are very encouraging. The conventional Backpropagation architecture, when extended to three internal levels instead of one, did not converge, whereas the comparable modular architecture (with twice as many connections, however) converged in a time comparable to that of the much smaller system with only one internal layer.

The main disadvantage of this scheme arises in the use of the network. In the case where a sensory input is provided and a motor response is desired, only one-half of the inputs to the abstract layer will be present. This places a much greater demand on the abstract layer to determine the correct pattern instead of an alternate. A more realistic assumption might be to assume that both sensory parts and motor parts of the pattern are present, and the pattern completion problem is no worse than in the feedforward system.

Having shown that modules of three-layer autoassociative systems can be built, we now ask whether error propagation at the internal level is really necessary. Consider that making the weight adjustment formula anisotropic actually improved convergence. Thus it seems plausible that a mixed learning system would be possible where the *output* weights are adjusted based on error correction but the *internal* weights are adjusted based on some other criterion. We conjecture the substitute criterion may only need to have certain smoothness properties and produce different internal states for the different input patterns. Thus there is likely to be a large family of learning algorithms that will work in the modular architecture.

Acknowledgements

This work was done at the program on Spin Glasses, Computation, and Neural Networks held during September to December 1986 at the Institute for Theoretical Physics, University of California at Santa Barbara, and organized by John Hopfield and Peter Young. This research was supported in part by NSF Grant No. PHY82-17853, supplemented by NASA. Thanks go to all the participants in the program who helped refine the ideas in this paper over numerous discussions, especially Jack Cowan, Christoph Koch, Alan Lapedes, Hanoch Gutfreund, Christoph von der Malsburg, Klaus Schulten, Sara Solla, and Haim Sompolinsky. In addition, at Rochester, Mark Fanty and Kenton Lynne provided helpful critiques. The importance of scaling problems in learning emerged during discussions at the Connectionist Summer School at Carnegie Mellon University in June 1986, particularly with Geoff Hinton. Thanks also go to Beth Mason and Peggy Meeker, who typed the many drafts of this manuscript.

References

Ackley, D.H., G.E. Hinton, and T.J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Science 9*, 1, 147-169, January-March 1985.

Barlow, H.B., "Single units and sensation: A neuron doctrine for perceptual psychology?," *Perception 1*, 371-394, 1972.

Baum, E., J. Moody, and F. Wilczek, "Internal representations for content addressable memory," Technical Report, Inst. for Theoretical Physics, U. California, Santa Barbara, December 1986.

Feldman, J.A., "Dynamic connections in neural networks," *Biological Cybernetics 46*, 27-39, 1982.

Lapedes, A. and R. Farber, "Programming a massively parallel, computation universal system: Static behavior," *Proc., Snowbird Conf. on Neural Nets and Computation*, April 1986.

Pearlmutter, B.A. and G.E. Hinton, "Maximization: An unsupervised learning procedure for discovering regularities," Technical Report, Carnegie Mellon U., May 1986.

Rumelhart, D.E. and D. Zipser, "Feature discovery by competitive learning," *Cognitive Science 9*, 1, 75-112, January-March 1985.

Rumelhart, D.E., G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation," in D.E. Rumelhart and J.L. McClelland.

(Eds). *Parallel Distributed Processing*. MIT Press, pp. 318-364, 1986.

Scalettar, R. and A. Zee, "A feedforward memory with decay," Technical Report NSF-ITP-86-118, Inst. for Theoretical Physics, U. California, Santa Barbara, 1986.

Zipser, D., "Programming networks to compute spatial functions," ICS Report 8608, Inst. for Cognitive Science, U. California, San Diego, La Jolla, June 1986.