

ASSOCIATIVE MEMORIES AND FEEDFORWARD NETWORKS: A SYNOPSIS OF NEURAL-NETWORK RESEARCH AT THE MILTON S. EISENHOWER RESEARCH CENTER

Neural networks have been of interest to the Mathematics and Information Science Group of the Milton S. Eisenhower Research Center for a number of years. Early efforts, which concentrated on associative memories and feedforward networks, are summarized as well as current and future work.

INTRODUCTION

Unlike the traditional approach to artificial intelligence in which symbolic processing is the dominant mechanism and little attention is paid to the underlying hardware, neural networks do not distinguish between hardware and software. The program and data are inextricably intertwined in a fashion that simulates in some respects the operation of a living brain. In particular, neural networks "do their computing" by the activation and inhibition of nodes that are somewhat similar to the presumed operation of neurons, or nerve cells. One might say that neural networks are an attempt to use the brain, rather than the computer, as a model for the mind.

One should not make too much of the analogy with biological systems, however. Rather, one should think of neural networks as inspired by neurophysiology and not as highly accurate simulations of the operation of a living system. Physiological plausibility is the key. Far too little is known of the detailed neurophysiology of individual neurons, let alone of their interconnections, to make any model of such a complex system much better than a good guess. Just as differences are found both within and between brains as well as both within and across species, differences are found, some subtle and some gross, between the many versions of neural networks. In addition, researchers vary in their emphasis of the importance of fidelity to biological systems: a psychologist or biologist is likely to be more concerned with physiological plausibility than a researcher using neural networks for an application.

Thus, no single model characterizes all neural networks, although some important classifications can be made. One involves the type of learning, namely, supervised or unsupervised. Hopfield networks and feedforward networks are examples of supervised learning, and competitive learning networks are examples of unsupervised learning. Another useful classification is functionality. Hopfield networks act as associative memories recalling exemplars (i.e., learned patterns) upon presentation of noisy or incomplete versions of the exemplars;

competitive learning networks can be used to detect intrastimulus correlations in the input without supervision; and feedforward networks learn to classify groups of input vectors with output (or target) vectors. We are interested in all three of these networks, but our research has concentrated on Hopfield-type and feedforward networks.

A good overview and bibliography of current neural-network technology, with some representative applications in the APL Fleet Systems Department, can be found in Ref. 1. The APL Space Department is developing recurrent, Hopfield-like networks and hardware implementations of them, as discussed in Ref. 2, and a scalable architecture for the implementation of a hetero-associative memory that resulted from this work is described in Ref. 3.

ASSOCIATIVE MEMORIES

Hopfield Networks

In a Hopfield network, the neurons (computing elements, nodes, or units) are simple bistable elements that can output either +1 (on) or -1 (off). The network is fully connected; that is, each neuron is connected to every other neuron except itself. Every neuron accepts input from its environment and passes output back to its environment. The equations in Figure 1 define a Hopfield network.

In its simplest use, a Hopfield network acts as an associative memory. When a noisy version of a stored exemplar is imposed or clamped to its initial state, it should eventually stabilize to the perfect exemplar. Hopfield⁴ showed that these networks can also be used to solve optimization problems, but this use will not be discussed here.

Hopfield networks have remarkable recall capabilities but are characterized by low storage capacities. Suppose that the n -dimensional exemplars $\mathbf{e}^1, \mathbf{e}^2, \dots, \mathbf{e}^p$ are to be stored in a Hopfield network. Now suppose the network is presented one of the exemplars (uncorrupted),

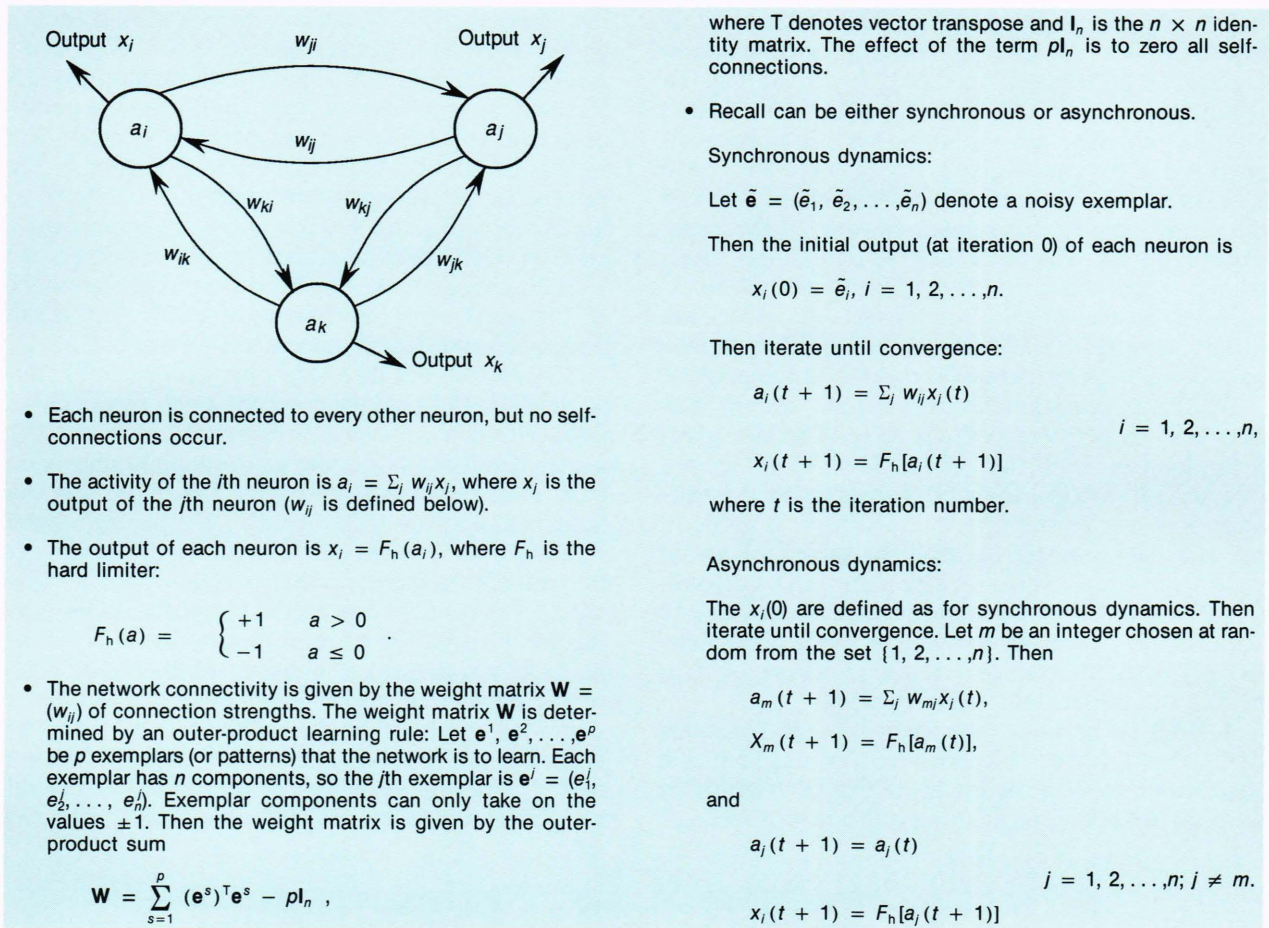


Figure 1. A Hopfield network and defining equations.

\mathbf{e}^r . Then, for synchronous updating, it is easy to show by using the equation that defines the weights (see Fig. 1) that the activity a_i of the i th neuron is

$$a_i = (n-1)e_i^r + \sum_{\substack{s=1 \\ s \neq r}}^p e_i^s \sum_{\substack{j=1 \\ j \neq i}}^n e_j^s e_j^r,$$

$$= S_i + N_i,$$

(1)

where S_i is the signal and N_i is the noise.

The output of the neuron is obtained by applying the hard-limiter function, $F_h(x)$. Thus, if $\text{sign}(N_i) = \text{sign}(S_i)$, or if $|S_i| > |N_i|$, $F_h(a_i) = e_i^r$; that is, the network recalls the i th component of \mathbf{e}^r . When the network is presented with an exemplar, perfect recall on the first iteration is desired. If the exemplars are stochastically independent, then the noise term will have a mean of zero and a variance σ^2 of $(n-1)(p-1)$, and, for large n and p , will follow a normal probability distribution. For this distribution, 99.5% of the probability lies within 3 standard deviations of the mean. To ensure that the signal term S_i satisfies $|S_i| \geq |N_i|$ with a probability of 0.995,

$$|S_i| = (n-1) \geq 3 \sqrt{(n-1)(p-1)}. \quad (2)$$

Thus,

$$n-1 \geq 9(p-1), \quad (3)$$

so that for a high probability of correct recall, the number of neurons must be about 9 times the number of stored patterns; or, conversely, the number of stored patterns must be no more than 11% of the number of neurons.

Now, consider a noisy version, $\tilde{\mathbf{e}}^r$, of the exemplar \mathbf{e}^r . Suppose $\tilde{\mathbf{e}}^r$ differs from \mathbf{e}^r in d elements. Then the expression for the activity a_i becomes

$$a_i \approx n(1-2\rho)e_i^r + N_i, \quad (4)$$

where $\rho = d/n$, that is, the fraction of elements of $\tilde{\mathbf{e}}^r$ that differs from \mathbf{e}^r . This equation is an approximation; the exact expression depends on whether or not \tilde{e}_i^r agrees with or differs from e_i^r . Thus, if $\tilde{\mathbf{e}}^r$ differs from \mathbf{e}^r in 25% of its elements, then

$$a_i \approx \frac{n}{2} e_i^r + N_i, \quad (5)$$

and a reasonably high probability still exists that the signal will exceed the noise.

The neuron-to-stored-pattern ratio of 9 derived in Equation 3 is in one sense too pessimistic, but in another sense it is more optimistic than is warranted. It is too pessimistic in that recall of the exemplar was required after only one iteration and with a very high probability. But it is too optimistic in that the exemplars were considered to be stochastically independent vectors and the vector input to the network was one of the exemplars. Exemplars of practical interest, for example, faces, numerals, and alphabetic characters, could not be expected to be stochastically independent and, in fact, are highly correlated. In addition, the network would normally have to deal with noisy versions of the exemplars.

For stochastically independent vectors, analysis indicates that Hopfield networks have good recall capacities as long as $p \leq 0.14n$. When the exemplars are not stochastically independent, the analysis becomes intractable, and one must resort to experiment. Consider a network that is taught to recognize subsets of the 10 digits from 0 to 9, which are represented as binary vectors of length 120. They are depicted in Figure 2 as 12×10 arrays so that their structure is easily discernible. A black square represents +1 and a white square represents -1 in the vector representation.

A Hopfield network was taught the first five digits (0 to 4), and its recall performance was measured as a function of noise in the exemplar. Figure 3 illustrates a typical recall sequence starting with a noisy version of

the exemplar "2." The noise parameter r is the fraction of bits in error; it was varied from 0 to 0.40 in increments of 0.05. For each of the five exemplars, 10 noisy versions for each value of r were used, and the performance on recall was averaged over the 10 trials. The results are shown as the black line in Figure 4. The recalled vector was considered correct if it differed by no more than 5 of the 120 bits of the corresponding exemplars. Good recall performance was found up to $r = 0.25$; thereafter, recall performance degraded rapidly. This result is not surprising because 25% noise results in a significant degradation of the exemplar.

The network was then taught the sixth exemplar, that is, the digit "5," and the experiment was repeated. Recall performance was greatly degraded (red line of Fig. 4). Even without noise, the network could only recall 83% of the exemplars correctly. This result was due largely to problems with the digit "2"; the high degree of correlation between the "2" and the "5" prevented the network from ever correctly recalling the "2," even with $r = 0$. As noise was added, performance degraded rapidly. Clearly, for these exemplars, the effective capacity of the network is about 5, which is only 4.2% of the number of neurons. Although this number is somewhat dependent on the exemplars (e.g., a network trained on six exemplars not including both the "2" and the "5" probably would have performed better), this value confirms the results of many experiments; that is, for

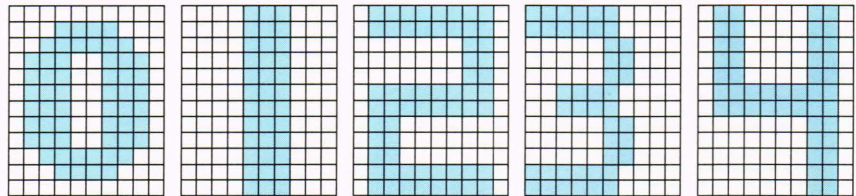


Figure 2. The 10 exemplars used to train the Hopfield networks.

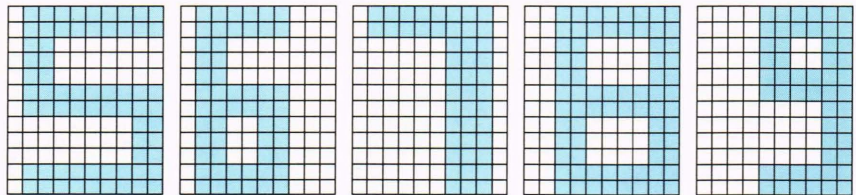
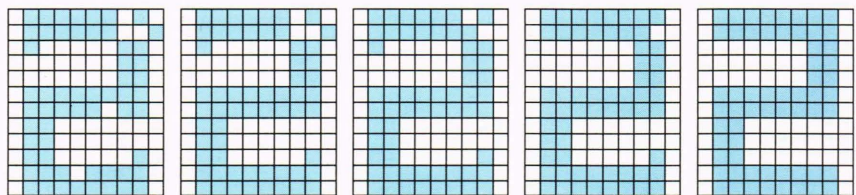
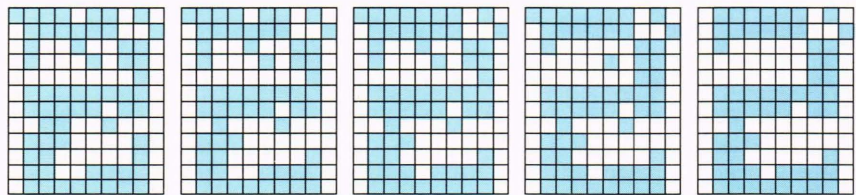


Figure 3. Recall of the exemplar "2" from a noisy version; asynchronous dynamics were used.



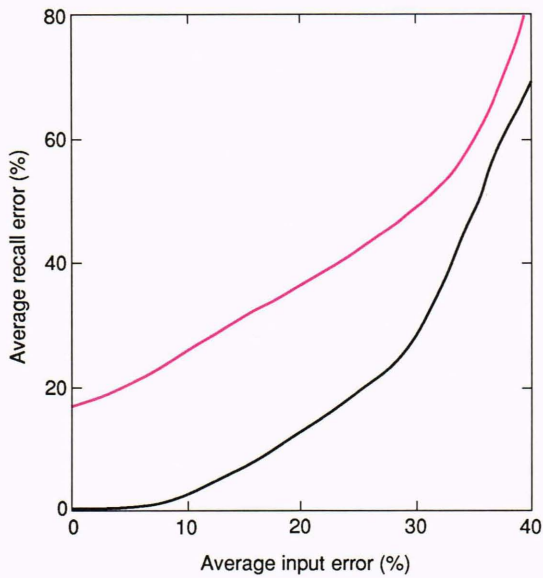


Figure 4. Average recall error as a function of average input error for a Hopfield network with 120 neurons. Two cases are illustrated: five learned exemplars (black) and six learned exemplars (red).

correlated exemplars, good recall performance requires that p not exceed 6% to 7% of the number of neurons.

An Optimal Hopfield Network

The storage capacity of a Hopfield network can be greatly increased if the learning rule of Figure 1 is replaced by a learning rule that minimizes the squared error E :

$$E = \frac{1}{2} \sum_{s=1}^p \sum_{l=1}^n (e_l^s - \sum_{k=1}^n w_{lk} e_k^s)^2 . \quad (6)$$

The equations that govern these networks are given in Figure 5.

Experiments were conducted with the 10 digits of Figure 2. All 10 digits were used as exemplars to be learned by the network. Recall performance of the network was tested as discussed in the preceding paragraphs. The results, again averaged over 10 different noisy exemplars for each r value, are shown in Figure 6. For comparison, the recall performance of a standard Hopfield network is included in the figure. The vastly improved performance of the optimal Hopfield network over that of the standard Hopfield network is evident.

The fault tolerance of the optimal Hopfield network is shown in Figure 7. In the figure, the recall performance of the network is given as synaptic weights are cut (i.e., set to zero). In this experiment the exemplars were not degraded. The results were obtained by randomly setting to zero $100 \times r\%$ of the weights as r was varied from 0 to 0.80 in increments of 0.05. It is rather remarkable that the network could recall perfectly the 10 exemplars with up to 60% of the weights set to zero.

- The learning rule for the optimal Hopfield network is based on minimizing the squared error E :

$$E = \frac{1}{2} \sum_s \sum_j (e_j^s - \sum_k w_{jk} e_k^s)^2 .$$
- Differentiating E with respect to w_{ij} results in the following iterative learning rule:

Initially, set w_{ij} to zero.

For $s = 1, 2, \dots, p$, iterate until convergence:

$$\Delta^s w_{ij} = \eta (e_i^s - \sum_k w_{ik} e_k^s) e_j^s \quad i = 1, 2, \dots, n; j = 1, 2, \dots, n.$$

$$w_{ij} = w_{ij} + \Delta^s w_{ij}$$

The learning rate parameter, η , is needed for numerical stability. A typical value of η is $1/n$.

Set the self-connection weights to zero:

$$w_{ij} = 0, i = 1, 2, \dots, n.$$
- The recall process can be either synchronous or asynchronous. In either case the equations that describe the process are identical to those given in Figure 1 for traditional Hopfield networks.

Figure 5. The learning equations for an optimal Hopfield network.

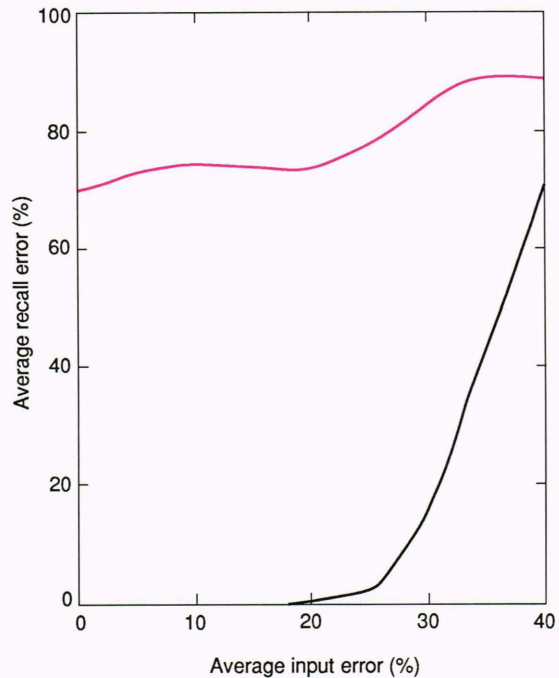


Figure 6. Average recall error as a function of average input error for an optimal Hopfield network (black) and for a traditional Hopfield network (red). Both networks have 120 neurons, and both learned the 10 exemplars of Figure 2.

BIDIRECTIONAL ASSOCIATIVE MEMORIES

The bidirectional associative memory (BAM) introduced by Kosko⁵ is the simplest form of a multilayer neural network. It is a two-layer neural network with bidirectional connections between the two layers; that

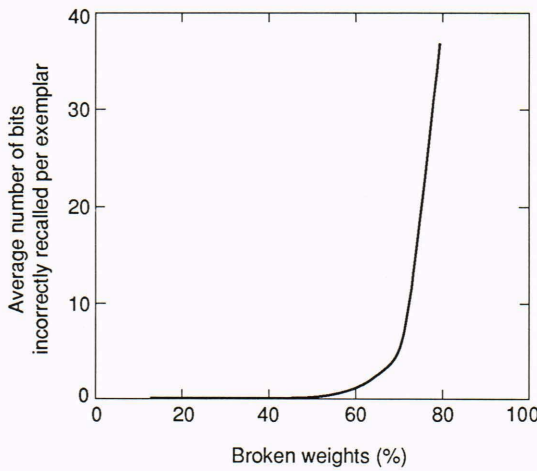


Figure 7. Average number of bits incorrectly recalled per exemplar as a function of the percentage of weights broken (set to zero) in a 120-neuron optimal Hopfield network. The exemplars learned by the network are those of Figure 2.

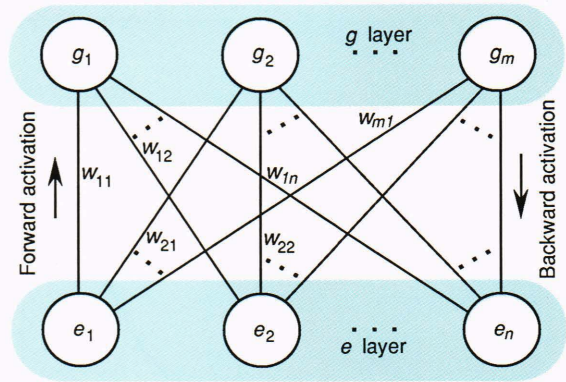
is, information flows forward and backward through the same connections. The BAM behaves as a heteroassociative memory. Each neuron in a BAM receives inputs from all neurons in the other layer but none from neurons in its own layer, as shown in Figure 8. Thus, for the same number of neurons, it has about one-fourth the connections of a Hopfield network.

Figure 9 shows the recall error in g as a function of the error rate in e obtained by numerical simulation. The simulations were done by using synchronous dynamics for a network with 100 neurons in the input layer and 100 neurons in the output layer. The number of stored pairs of vectors was 10, and the results shown in Figure 9 are the average recall errors for samples of 400 random vectors. In the figure, the recall error is shown after the first forward flow of information and after the network has come to a steady state.

Simulations were also carried out to study the performance of a BAM for various values of p , the number of stored pairs of vectors. Figure 10 shows the recall error of a BAM with 100 input and 100 output neurons versus p for three levels of input noise. The errors are the average values obtained from samples of 400 random vectors. A good recall is defined as one that has an error not larger than 1.0%. For the case described in Figure 10, a good recall is obtained for $p \leq 10$.

OPTIMAL ASSOCIATIVE MEMORY

Although the BAM has a larger memory capacity than a Hopfield network per number of connections, it still has relatively low memory capacity and small basins of attraction for the stored patterns. We have investigated two-layer networks in which the single integer-element BAM weight matrix is replaced by a pair of continuous-element matrices that give the best mapping, in the least-squares sense, for both the forward and backward directions. We refer to this network as an optimal associative memory (OAM) (Fig. 11). The OAM is a two-layer network that, on the basis of experiments, has a larger



- Every node in the bottom layer is connected to every node in the top layer, and, similarly, every node in the top layer is connected to every node in the bottom layer. No intralayer connections exist.

- The activity of the i th unit is $a_i = \sum_j w_{ij}x_j$, where x_j is the output of the j th unit.

- The output of each unit is $x_i = F_h(a_i)$, where F_h is the hard limiter:

$$F_h(a) = \begin{cases} +1 & a > 0 \\ -1 & a \leq 0 \end{cases}$$

- The learning rule that determines the connection strength matrix, or weight matrix $\mathbf{W} = (w_{ij})$, is the outer-product learning rule: Let $(\mathbf{e}^1, \mathbf{g}^1), (\mathbf{e}^2, \mathbf{g}^2), \dots, (\mathbf{e}^p, \mathbf{g}^p)$ be p pairs of exemplars or patterns that the network is to learn to associate. The e -type exemplars have n components, and the g -type exemplars have m components. All components are bipolar, that is, either $+1$ or -1 . Then, the $n \times m$ weight matrix is given by the outer-products sum

$$\mathbf{W} = \sum_{s=1}^p (\mathbf{e}^s)^T \mathbf{g}^s,$$

where T denotes vector transpose.

- As is the case with Hopfield networks, the recall process can be either synchronous or asynchronous. We have used only synchronous recall in our work. Let $\tilde{\mathbf{e}} = (\tilde{e}_1, \tilde{e}_2, \dots, \tilde{e}_n)$ and $\tilde{\mathbf{g}} = (\tilde{g}_1, \tilde{g}_2, \dots, \tilde{g}_m)$ denote noisy exemplars of the e - and g -types, respectively. The initial output of the network is

$$x_i^{(e)}(0) = \tilde{e}_i, \quad i = 1, 2, \dots, n,$$

$$x_j^{(g)}(0) = \tilde{g}_j, \quad j = 1, 2, \dots, m.$$

Then iterate until convergence:

$$a_i^{(e)}(t+1) = \sum_j w_{ij}x_j^{(g)}(t) \quad i = 1, 2, \dots, n,$$

$$x_i^{(e)}(t+1) = F_h[a_i^{(e)}(t+1)]$$

and

$$a_j^{(g)}(t+1) = \sum_i x_i^{(e)}(t)w_{ij} \quad j = 1, 2, \dots, m.$$

$$x_j^{(g)}(t+1) = F_h[a_j^{(g)}(t+1)]$$

Figure 8. A BAM and defining equations.

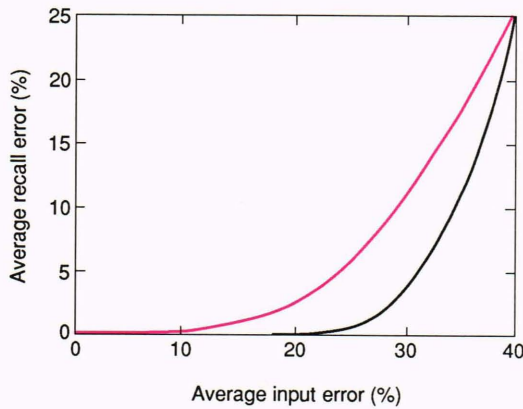


Figure 9. Average recall error as a function of average input error for a BAM with 100 input and 100 output neurons after the first forward flow (red) and at steady state (black). The number of stored pairs of vectors is 10. The output error is the average over 400 random vectors.

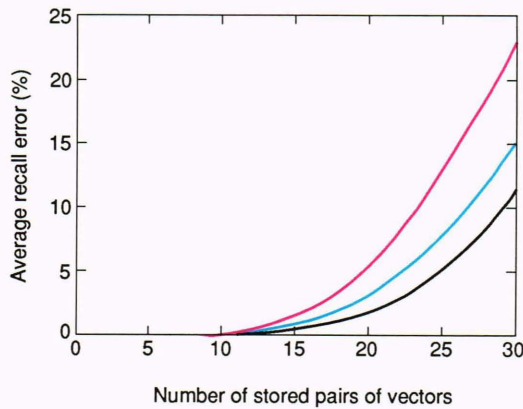


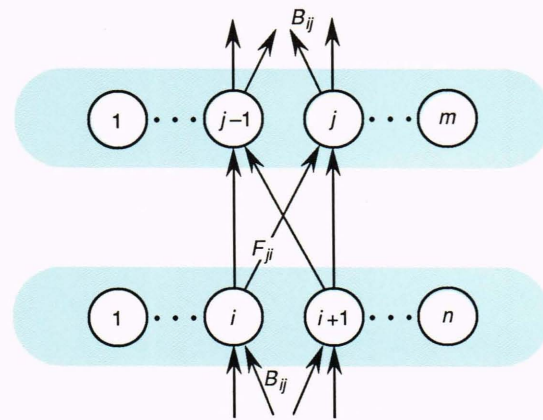
Figure 10. Average recall error for a BAM as a function of the number of stored vectors for the following input noise levels (%): 0 (black), 10 (blue), and 20 (red). The BAM has 100 input and 100 output neurons, and the results are averages for samples of 400 random vectors.

memory capacity and larger radii of attraction than a BAM with the same number of neurons in each layer, but twice as many connections.

We performed numerical simulations of the OAM to demonstrate its memory capacity and to compare it with that of the BAM. The initial values of the connection matrices were all set to zero and were then modified by using the learning rules of Figure 11.

Pairs of prototype vectors (\mathbf{e}^s , \mathbf{g}^s) were presented to the network until the connection matrices \mathbf{F} and \mathbf{B} converged. Results for a network of 50 input neurons and 50 output neurons are shown in Figure 12 and compared with those of the BAM. The number of iterations (presentations of the prototype patterns) needed for convergence of the OAM matrices ranged from a few dozen to a few hundred (the number usually increases with the number of stored pairs of vectors).

The BAM has good recall (for zero-noise input vectors) if the number of stored pairs of vectors is six or fewer. The OAM, on the other hand, has good recall for up to



- The forward and backward connection matrices (\mathbf{F} and \mathbf{B}) of the OAM are not transpose pairs, as is the case with a BAM; therefore, the number of connections in an OAM is twice that of a BAM with the same number of neurons in the two layers. As in the BAM, each neuron in an OAM receives inputs from all neurons in the other layer but none from neurons in its own layer. The operation of an OAM is identical to that of a BAM, except for the learning rule. As with the BAM, let $(\mathbf{e}^1, \mathbf{g}^1), (\mathbf{e}^2, \mathbf{g}^2), \dots, (\mathbf{e}^p, \mathbf{g}^p)$ be p association pairs of exemplars the OAM is to learn.

- Learning is based on minimizing the two squared errors:

$$E_f = \frac{1}{2} \sum_s \sum_j (g_j^s - \sum_k f_{jk} g_k^s)^2,$$

$$E_g = \frac{1}{2} \sum_s \sum_i (e_i^s - \sum_k f_{ik} e_k^s)^2.$$

- The connection matrix \mathbf{F} is obtained by letting the network connections, f_{ij} , evolve along trajectories that are opposite in direction to the gradient, $\partial E_g / \partial f_{ij}$, of E_g . The learning rule that results is

$$\Delta^s f_{ij} = \eta_g (g_i^s - \bar{g}_i^s) e_j^s.$$

Here, $\Delta^s f_{ij}$ is the change to be made in f_{ij} following a presentation of the input vector \mathbf{e}^s , g_i^s is the target output for neuron i of the output layer, and \bar{g}_i^s is the actual input to this neuron (i.e., $\bar{g}_i^s = \sum_j f_{ij} e_j^s$).

- In matrix notation this can be rewritten as

$$\Delta^s \mathbf{F} = \eta_g (\mathbf{g}^s - \bar{\mathbf{g}}^s) \mathbf{e}^s$$

- The same arguments apply to the backward $\mathbf{g} \rightarrow \mathbf{e}$ flow. The learning rule for the backward connection matrix is

$$\Delta^s \mathbf{B} = \eta_e (\mathbf{e}^s - \bar{\mathbf{e}}^s) \mathbf{g}^s.$$

Figure 11. An OAM and defining equations.

about 50 stored pairs (not shown in Figure 12). In other words, the memory capacity of the OAM is about 8 times larger than that of the BAM for the simulated network. For noisy patterns, the memory capacity of the OAM is about 3 times that of the BAM, as can be seen in Figure 12. This research is covered in greater detail by Schneider and Sigillito.⁶

FEEDFORWARD NETWORKS

Much of our current research is concerned with feedforward networks that use the back-propagation learning algorithm. Feedforward networks are composed of an input layer, one or more intermediate or hidden lay-

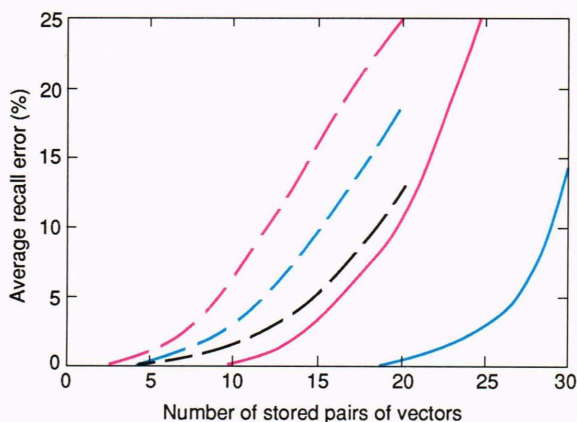


Figure 12. Average output recall error as a function of the number of stored pairs of vectors for a BAM (dashed lines) and OAM (solid lines) with 50 neurons in the input layer and 50 neurons in the output layer. The following input noise levels were used (%): 0 (black), 10 (blue), and 20 (red). The results were obtained by using random samples of 200 vectors. For the OAM, the average recall error was zero at 0% input noise for the number of stored pairs of vectors shown.

ers, and an output layer. All units in a given layer are connected to all units in the layer above. No other connections exist (Fig. 13). (Many other connection topologies could be used. For example, all nodes at a given layer could be connected to all nodes at all higher levels. Neither these fully connected networks nor other connection topologies will be considered here.) Input units do not perform computations but serve only to distribute the input to the computing units (neurons) in the hidden layer. Units in the hidden layer are not connected to the outside world, but after processing their inputs, they pass the results to the neurons in the output layer.

In addition to the feedforward process just described, a back-propagation learning process is used: errors in the output are propagated backward from the output layer to the hidden layer and from the hidden layer to the input layer. In this learning process, the connection weights are modified so as to minimize the total error in the output of the network.

Given a set \mathbf{I} of input vectors, a set \mathbf{T} of target vectors, and an unknown map $\mathbf{M}:\mathbf{I}\rightarrow\mathbf{T}$, neural networks are used to learn an approximation \mathbf{M}_a of \mathbf{M} . Thus, suppose $\mathbf{I}_s = (\mathbf{I}^1, \mathbf{I}^2, \dots, \mathbf{I}^p)$ is a subset of \mathbf{I} , and $\mathbf{T}_s = (\mathbf{T}^1, \mathbf{T}^2, \dots, \mathbf{T}^p)$ is a subset of \mathbf{T} . Then \mathbf{M}_a is the mapping that results by teaching the network the p associations: $\mathbf{I}^i \rightarrow \mathbf{T}^i$, $i = 1, 2, \dots, p$. The pairs $(\mathbf{I}^i, \mathbf{T}^i)$ for $i = 1, 2, \dots, p$ are called the training pairs. During the learning phase, the network adaptively changes its synaptic weights to effect the map \mathbf{M}_a and builds an internal representation of the mapping. For the network to learn arbitrary mappings, the use of hidden layers is essential.⁷

In one effort, feedforward networks were taught to perform a discrimination task that involved radar returns from the ionosphere. We did this work in collaboration with the APL Space Physics Group. The work is dis-

cussed in the article by Sigillito et al. elsewhere in this issue.

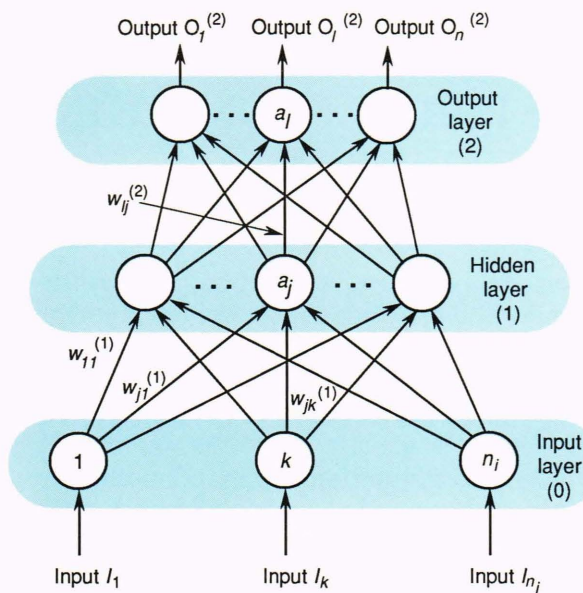
Another project that was recently completed involved the use of a feedforward network to predict the onset of diabetes mellitus from clinical evidence.⁸ This research used the networks to classify female subjects into two classes (high or low probability of developing diabetes within 5 years) on the basis of eight clinical variables: number of times pregnant, plasma glucose concentration after 2 h in an oral glucose tolerance test, diastolic blood pressure, triceps skin-fold thickness, 2-h serum insulin, body mass index, age, and diabetes pedigree function. The diabetes pedigree function was developed to provide a history of diabetes mellitus in relatives and the genetic relationship of those relatives to the subject.⁹ It provides a measure of the expected genetic influence of affected and unaffected relatives on the subject's eventual diabetes risk. The network was trained on data from 576 subjects and was tested on 192 different subjects from the same population. Performance on the testing set reached a high of 84% correct. Multiple linear regression techniques that are the standard for epidemiological studies such as this performed at the 76% level.

Several research projects are being pursued. One is motivated by Georgopoulos et al.¹⁰ at the JHU School of Medicine. They have developed a model of the motor cortex that simulates a monkey making a mental rotation of an object in space before performing a learned sensorimotor task. The model is based on single-cell recordings from the motor cortex, and the analysis is primarily statistical in nature. We developed a model that makes many of the same predictions, but it is based on the output of a feedforward neural network. We are working with Georgopoulos and his colleagues to develop a model to simulate projections from the motor cortex to the motor neurons themselves.

Another project, inspired by the work of Qian and Sejnowski,¹¹ is concerned with using neural networks to predict the secondary structure of proteins on the basis of their primary amino acid structure. Qian and Sejnowski¹¹ produced results that were better than those obtained by using traditional methods, and later we, in collaboration with Harry R. Goldberg of the Department of Biophysics at the JHU School of Medicine, obtained slightly improved results. Nevertheless, they were not good enough for practical use. We are exploring the use of recurrent networks to improve this work.

Future work includes both applied and basic research. One applied project will be concerned with the use of neural networks to perform process control tasks in a manufacturing environment, and a second will be concerned with the use of neural networks to perform inspection tasks. Both of these projects are in the planning stages.

In basic research, we are investigating the properties of a particular class of recurrent networks that are candidates for modeling the dynamic development of human and animal cognitive maps. Interpretation of our early results suggests that our network exhibits behavior



- Let the vector $\mathbf{I} = (I_1, I_2, \dots, I_{n_i})$ denote input to the network. The corresponding output that the network is to learn will be denoted by the vector $\mathbf{T} = (T_1, T_2, \dots, T_{n_o})$. Here, n_i is the number of input nodes and n_o is the number of output nodes.
- The output, $O_i^{(0)}$, of the i th input node is simply its input, that is, $O_i^{(0)} = I_i$.
- The activity of the j th node in the hidden layer (layer 1) is

$$a_j^{(1)} = \sum_i w_{ji}^{(1)} O_i^{(0)},$$

Figure 13. A feedforward network and the defining equations.

consistent with habit formation, goal selection, and sensitivity to mismatches between an anticipated and actual environmental event, even though it was not explicitly trained to show any of these behavioral traits. These results are especially promising because they support the possibility of developing cognitive control systems within a neural-network environment.

REFERENCES

- Roth, M. W., "Neural-Network Technology and Its Applications," *Johns Hopkins APL Tech. Dig.* **9**, 242-253 (1988).
- Jenkins, R. E., "Neurodynamic Computing," *Johns Hopkins APL Tech. Dig.* **9**, 232-241 (1988).
- Boahen, K. A., Pouliquen, P. O., Andreou, A. G., and Jenkins, R. E., "A Heteroassociative Memory Using Current-Mode MOS Analog VLSI Circuits," *IEEE Trans. Circuits Syst.* **36**, 747-755 (1989).
- Hopfield, J. J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," in *Proc. National Academy of Sciences, USA* **79**, pp. 2554-2558 (1982).
- Kosko, B., "Adaptive Bidirectional Associative Memories," *Appl. Opt.* **26**, 4947-4960 (1987).
- Schneider, A., and Sigillito, V. G., "Two-Layer Binary Associative Memories," in *Progress in Neural Networks*, Ablex Publishing Co., Norwood, N.J. (in press).
- Rumelhart, D. E., and McClelland, J. L., eds., in *Parallel Distributed Processing*, Vol. 1, MIT Press, Cambridge, Mass., pp. 318-364 (1986).
- Hutton, L. V., Sigillito, V. G., and Johannes, R. S., "An Interaction Between Auxiliary Knowledge and Hidden Nodes on Time to Convergence," in *Proc. 13th Symposium on Computer Applications in Medical Care* (in press).
- Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., and Johannes, R. S., "Using the ADAP Learning Algorithm to Forecast the Onset

and the output of this node is

$$O_i^{(1)} = f(a_i^{(1)}),$$

where

$$f(x) = 1/(1 + e^{-x}).$$

- Similar equations apply to the output layer (layer 2):

$$a_j^{(2)} = \sum_i w_{ji}^{(2)} O_i^{(1)},$$

$$O_i^{(2)} = f(a_i^{(2)}).$$

Here, the sum is over the n_h hidden nodes.

- The learning equations result from minimizing the squared error by using standard steepest-descent techniques:

$$E = 1/2 \sum_j (T_j - O_j^{(2)})^2,$$

where the sum is over the n_o output nodes. These equations are

$$w_{ji}^{(2)}(t+1) = w_{ji}^{(2)}(t) + \eta \delta_j^{(2)} O_i^{(1)} + \alpha \Delta w_{ji}^{(2)}(t),$$

where

$$\delta_j^{(2)} = (T_j - O_j^{(2)}) O_j^{(2)} (1 - O_j^{(2)}).$$

The constant $\eta < 1$ is a learning rate parameter, and the last term, $\alpha \Delta w_{ji}^{(2)}(t)$ (α is a positive constant), is a momentum term that is used to suppress high-frequency fluctuations in the weights during learning. Similar equations govern the change to $w_{ji}^{(1)}$:

$$w_{ji}^{(1)}(t+1) = w_{ji}^{(1)}(t) + \eta \delta_j^{(1)} O_i^{(0)} + \alpha \Delta w_{ji}^{(1)}(t),$$

where

$$\delta_j^{(1)} = O_j^{(1)} (1 - O_j^{(1)}) \sum_m \delta_m^{(2)} w_{mj}^{(2)}.$$

of Diabetes Mellitus," in *Proc. 12th Symposium on Computer Applications in Medical Care*, pp. 261-265 (1988).

¹⁰ Georgopoulos, A. P., Schwartz, A. B., and Kettner, R. E., "Neuronal Population Coding of Movement Direction," *Science* **233**, 1416-1419 (1986).

¹¹ Qian, N., and Sejnowski, T. J., "Predicting the Secondary Structure of Globular Proteins Using Neural Network Models," *J. Mol. Biol.* **202**, 865-884 (1988).

THE AUTHOR



VINCENT G. SIGILLITO received a B.S. degree in chemical engineering (1958), and M.A. (1962) and Ph.D. (1965) degrees in mathematics from the University of Maryland. He joined APL in 1958 and is supervisor of the Mathematics and Information Science Group in the Milton S. Eisenhower Research Center. Dr. Sigillito is also chairman of the Computer Science Program Committee of the G.W.C. Whiting School of Engineering's Continuing Professional Programs, and has served as interim Associate Dean of the Continuing Professional Programs.