Neural Networks, Vol. 2, pp. 243–257, 1989
Printed in the USA.

0893-6080/89 $3.00 + .00

REVIEW ARTICLE

Neural Network Models for Pattern Recognition and Associative Memory

GAIL A. CARPENTER

Northeastern University and Boston University

(*Received* 25 *January* 1989; *revised and accepted* 22 *February* 1989)

Abstract—*This review outlines some fundamental neural network modules for associative memory, pattern recognition, and category learning. Included are discussions of the McCulloch-Pitts neuron, perceptrons, adaline and madaline, back propagation, the learning matrix, linear associative memory, embedding fields, instars and outstars, the avalanche, shunting competitive networks, competitive learning, computational mapping by instar/outstar families, adaptive resonance theory, the cognitron and neocognitron, and simulated annealing. Adaptive filter formalism provides a unified notation. Activation laws include additive and shunting equations. Learning laws include back-coupled error correction, Hebbian learning, and gated instar and outstar equations. Also included are discussions of real-time and off-line modeling, stable and unstable coding, supervised and unsupervised learning, and self-organization.*

Keywords—Neural network, Perceptron, Associative memory, Adaptive filter, Pattern recognition, Competition, Category formation, Self-organization.

1. INTRODUCTION

Neural network analysis exists on many different levels. At the highest level (Figure 1) we study theories, architectures, hierarchies for big problems such as early vision, speech, arm movement, reinforcement, cognition. Each architecture is typically constructed from pieces, or *modules*, designed to solve parts of a bigger problem. These pieces might be used, for example, to associate pairs of patterns with one another or to sort a class of patterns into various categories. In turn, for every such module there is a bewildering variety of examples, equations, simulations, theorems, and implementations, studied under various conditions such as fast or slow input presentation rates, supervised or unsupervised learning, real-time or off-line dynamics. These variations and their applications are now the subject of hundreds of talks and papers each year. In this review I will focus on the middle level, on some of the fundamental neural network modules that carry out associative memory, pattern recognition, and category learning.

Even then this is a big subject. To help organize it further I will trace the historical development of the main ideas, grouped by theme rather than by strict chronological order. But keep in mind that there is a much more complex history, and many more contributors, than you will read about here. I refer you to the Bibliography section, in particular to the recent collection of articles in the book *Neurocomputing: Foundations of Research*, edited by James A. Anderson and Edward Rosenfeld (1988).

Acknowledgements: This article is based upon a tutorial lecture given on 6 September 1988, at the First Annual Meeting of the International Neural Network Society, in Boston, Massachusetts. The author's research is supported in part by grants from the Air Force Office of Scientific Research (AFOSR F49620-86-C-0037 and AFOSR F49620-87-C-0018) and the National Science Foundation (NSF DMS-86-11959). I wish to thank these agencies for their long-term support of neural network research, and also to thank my colleagues at the Boston University Center for Adaptive Systems for their generous ongoing contributions of knowledge, skills, and friendship.

Requests for reprints should be sent to Prof. Gail A. Carpenter, Center for Adaptive Systems, Boston University, 111 Cummington Street, Boston, MA 02215, USA.

2. THE McCULLOCH-PITTS NEURON

We would probably all agree to begin with the McCulloch-Pitts neuron (Figure 2a). The McCulloch-Pitts model describes a neuron whose activity x_i is the sum of inputs that arrive via weighted pathways. The input from a particular pathway is an incoming signal S_i multiplied by the weight w_{ji} of that pathway. These weighted inputs are summed independently. The outgoing signal $S_j = f(x_j)$ is typically a nonlinear

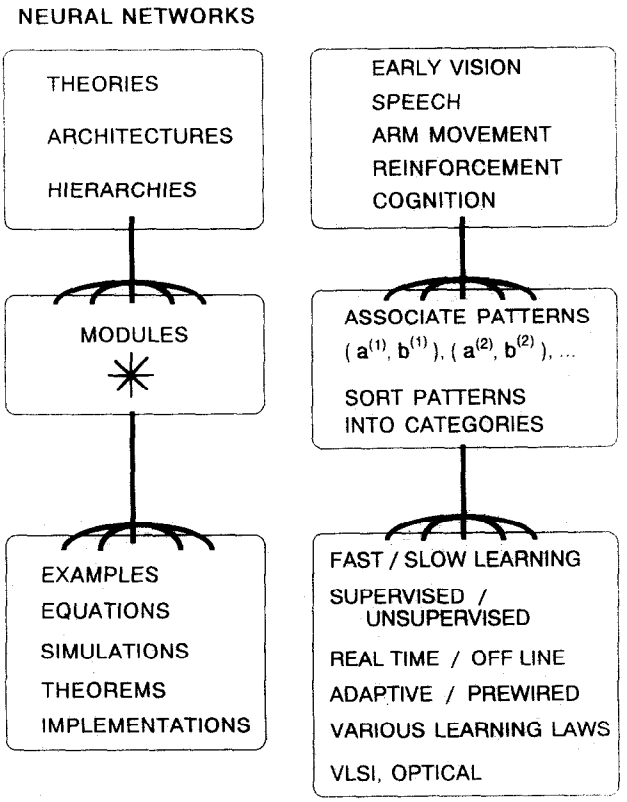


FIGURE 1. Levels of neural network analysis.

function—binary, sigmoid, threshold-linear—of the activity x_j in that cell. The McCulloch-Pitts neuron can also have a bias term θ_j, which is formally equivalent to the negative of a threshold of the outgoing signal function.

3. ADAPTIVE FILTER FORMALISM

There is a very convenient notation for describing the McCulloch-Pitts neuron, called the *adaptive filter*. It is this notation that I will use to translate models into a common language so that we can compare and contrast them. The elementary adaptive filter depicted in Figure 2b has:

1. a level F_1 that registers an input pattern vector;
2. signals S_i that pass through weighted pathways; and
3. a second level F_2 whose activity pattern is here computed by the McCulloch-Pitts function:

$$x_j = \sum_i S_i w_{ij} + \theta_j. \tag{1}$$

The reason that this formalism has proved so extraordinarily useful is that the F_2 level of the adaptive filter computes a pattern match, as in eqn (2):

$$\sum_i S_i w_{ij} = \mathbf{S} \cdot \mathbf{w}_j = \|\mathbf{S}\| \, \|\mathbf{w}_j\| \cos(\mathbf{S}, \mathbf{w}_j). \tag{2}$$

The independent sum of the weighted pathways in (2) equals the dot product of the signal vector **S** times the weight vector $\mathbf{w}_j$. This term can be factored into the "energy," the product of the lengths of **S** and $\mathbf{w}_j$; times a dimensionless measure of "pattern match," the cosine of the angle between the two vectors. Suppose that the weight vectors $\mathbf{w}_j$ are normalized and the bias terms θ_j are all equal. Then the activity vector **x** across the second level describes the degree of match between the signal vector **S** and the various weighted pathway vectors $\mathbf{w}_j$: the F_2 node with the greatest activity indicates the weight vector that forms the best match.

4. LOGICAL CALCULUS AND INVARIANT PATTERNS

The paper that first describes the McCulloch-Pitts model is entitled "A logical calculus of the ideas immanent in nervous activity" (McCulloch & Pitts, 1943). In that paper, McCulloch and Pitts analyze the adaptive filter *without adaptation*. In their models, the weights are constant. There is no learn-

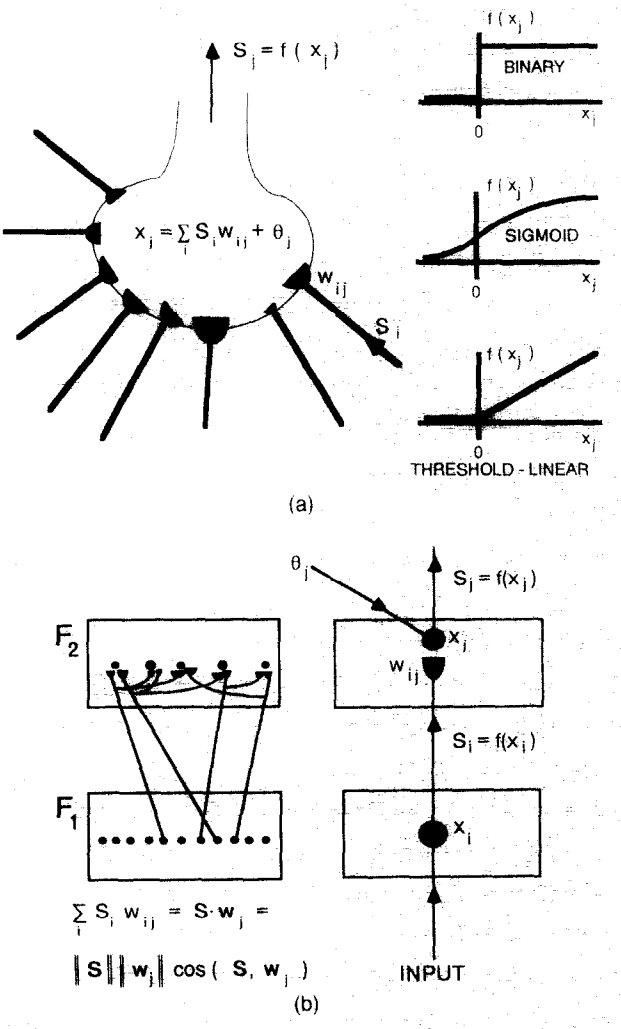


FIGURE 2. The McCulloch-Pitts model (a) as a neuron, with typical nonlinear signal functions; (b) as an adaptive filter.

ing. The 1943 paper shows that given the linear filter with an absolute inhibition term:

$$x_i = \sum_i S_i w_{ij} + \theta_j - [\text{inhibition}] \qquad (3)$$

and binary output signals, these networks can be configured to perform arbitrary logical functions. And if you are looking for applications of neural network research you need only read the memoires of John von Neumann (1958) to see how heavily the McCulloch-Pitts formalism influenced the development of present-day computer architectures.

In a sense this was looking backwards, to the early 20th century mathematics of *Principia Mathematica* (Russell & Whitehead, 1910, 1912, 1913). A glance at the 1943 McCulloch-Pitts paper shows that it is written in notation with which few of us are now familiar. (This is a good example of revolutionary ideas being expressed in the language of a previous era. As the revolution comes about a new language evolves, making the seminal papers "hard to read.") McCulloch and Pitts also clearly looked forward toward present day neural network research. For example, a later paper is entitled "How we know universals: The perception of auditory and visual forms" (Pitts & McCulloch, 1947). There they examine ideas in pattern recognition and the computation of invariants. They thus took their research program into a domain distinctly different from the earlier analysis of formal network groupings and computation. Still they considered only models without learning.

5. PERCEPTRONS AND BACK-COUPLED ERROR CORRECTION

The McCulloch-Pitts papers were extraordinarily influential, and it was not long before the next generation of researchers added learning and adaptation. One great figure of the next decade was Frank Rosenblatt, whose name is tied with the perceptron model (Rosenblatt, 1958). Actually, "perceptron" refers to a large class of neural models. The models that Rosenblatt himself developed and studied are numerous and varied; see, for example, his book, *Principles of Neurodynamics* (1962).

The core idea of the perceptron is the incorporation of learning into the McCulloch-Pitts neuron model. Figure 3 illustrates the main elements of the perceptron, including, in Rosenblatt's terminology, the sensory unit (S); the association unit (A), where the learning takes place; and the response unit (R).

One of the many perceptrons that Rosenblatt studied, one that remains important to the present day, is the *back-coupled perceptron* (Rosenblatt, 1962, section IV). Figure 4a illustrates a simple version of the back-coupled perceptron model, with a

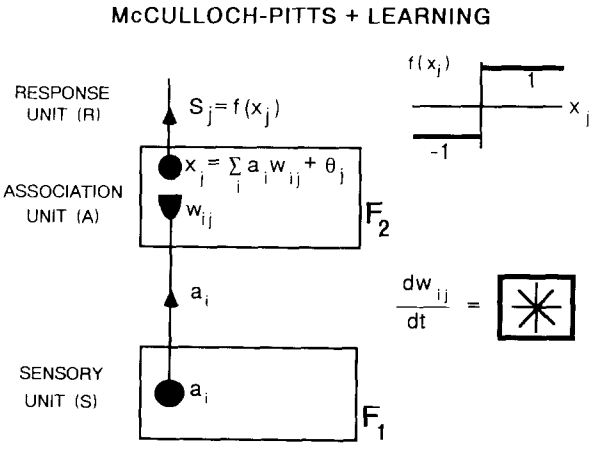


FIGURE 3. Principal elements of a Rosenblatt perceptron: sensory unit (S), association unit (A), and response unit (R).

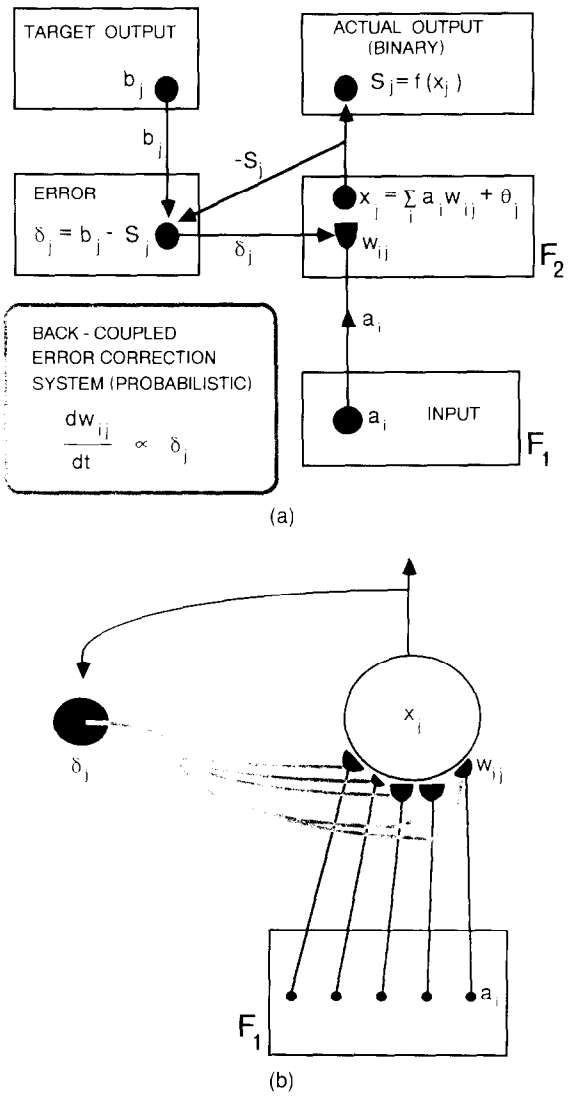


FIGURE 4. Back-coupled error correction. (a) The difference between the target output and the actual output is fed back to adjust weights when an error occurs. (b) All weights w_{ij} fanning in to the jth node are adjusted in proportion to the error δ_j at that node.

feedforward adaptive filter and binary output signal. Weights w_{ij} are adapted according to whether the actual output S_j matches a target output b_j imposed on the system. The actual output vector is subtracted from the target output vector; their difference is defined as the error; and that difference is then fed back to adjust the weights, according to some probabilistic law. Rosenblatt called this process *back-coupled error correction*. It was well known at the time tht these two-level perceptrons could sort linearly separable inputs, which can be separated by a hyperplane in vector space, into two classes. Figure 4b shows back-coupled error correction in more detail. In particular the error δ_j is fed back to every one of the weights converging on the jth node.

6. ADALINE AND MADALINE

Research in the 1960s did not stop with these two-level perceptrons, and continued on to multiple-level perceptrons, as indicated below. But first let us consider another development that took place shortly after Rosenblatt's perceptron formulations. This is the set of models used by Bernard Widrow and his colleagues, especially the *adaline and madaline* perceptrons. The adaline model has just one neuron in the F_2 level in Figure 5; the madaline, or many-adaline, model has any number of neurons in that level. Figure 5 highlights the principal difference between the adaline/madaline and Rosenblatt's two-level feedforward perceptron: an adaline/madaline model compares the *analog* output x_j with the target output b_j. This comparison provides a more subtle index of error than a law that compares the *binary* output

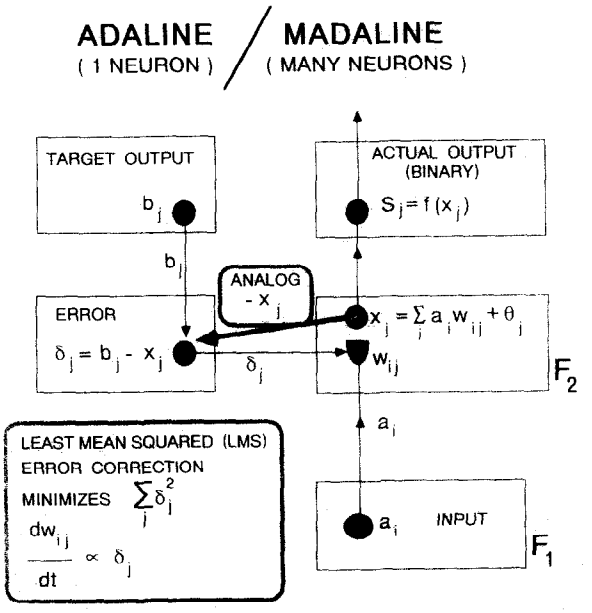


FIGURE 5. The adaline and madaline perceptrons use the analog output x_j, rather than the binary output S_j, in the back-coupled error correction procedure.

with the target output. The error $b_j - x_j = \delta_j$ is fed back to adjust weights using a Rosenblatt back-coupled error correction rule:

$$\frac{dw_{ij}}{dt} = \alpha\delta_j \frac{a_i}{|a|} \tag{4}$$

This rule minimizes the mean squared error:

$$\sum \delta_j^2 \tag{5}$$

averaged over all inputs (Widrow & Hoff, 1960). It is therefore known as the *least mean squared* error correction rule, or LMS.

Once again, adaline and madaline provide many examples of the technological spinoffs already generated by neural network research. Some of these are summarized in a recent article (Widrow & Winter, 1988) in a *Computer* special issue on artificial neural systems. There the authors describe adaptive equalizers and adaptive echo cancellation in modems, antennae, and other engineering applications, all directly traceable to early neural network designs.

7. MULTILEVEL PERCEPTRONS: EARLY BACK PROPAGATION

We have so far been talking only about two-level perceptrons. Rosenblatt, not content with these, also studied multilevel perceptrons, as described in *Principles of Neurodynamics*. One particularly interesting section in that book is entitled "Back-propagating error correction procedures." The back-propagation model described in that section anticipates the currently used back-propagation model, which is also a multilevel perceptron. In chapter 13, Rosenblatt defines a back-propagation algorithm that has, like most of his algorithms, a probabilistic learning law; he proves a theorem about this system; and he carries out simulations. His chapter, "Summary of three-layer series-coupled systems: Capabilities and deficiencies," is equally revealing. This chapter includes a hard look at what is lacking as well as what is good in Rosenblatt's back-propagation algorithm, and it puts the lie to the myth that all of these systems were looked at only through rose-colored glasses.

8. LATER BACK PROPAGATION

Let us now move on to what has become one of the most useful and well-studied neural network algorithms, the model we now call back propagation. This system was first developed by Paul Werbos (1974), as part of his Ph.D. thesis "Beyond regression: New tools for prediction and analysis in the behavioral sciences"; and independently discovered by David Parker (1982). (See Werbos (1988) for a review of the history of the development of back propagation.)

The most popular back-propagation examples carry out associative learning: during training, a vector pattern **a** is associated with a vector pattern **b**; and subsequently **b** is recalled upon presentation of **a** (Rumelhart, Hinton, & Williams, 1986). The back-propagation system is trained under conditions of *slow learning*, with each pattern pair (**a**, **b**) presented repeatedly during training. The basic elements of a typical back-propagation system are the McCulloch-Pitts linear filter with a sigmoid output signal function and Rosenblatt back-coupled error correction. Figure 6 shows a block diagram of a back-propagation system that is a three-level perceptron. The input signal vector converges on the "hidden unit" F_2 level after passing through the first set of weighted pathways w_{ij}. Signals S_j then fan out to the F_3 level, which generates the actual output of this feedforward system. A back-coupled error correction system then compares the actual output S_k with a target output b_k and feeds back their difference to all the weights w_{jk} converging on the kth node. In this process the difference $b_k - S_k$ is also multiplied by another term, $f'(x_k)$, computed in a "differentiator" step. One function of this step is to ensure that the weights remain in a bounded range: the shape of the sigmoid signal function implies that weights w_{jk} will stop growing if the magnitude of the activity x_k becomes too large, since then the derivative term $f'(x_k)$ goes to zero. Then there is a second way in which the error correction is fed back to the lower level. This is where the term "back propagation" enters: the weights w_{jk} in the feedforward pathways from F_2 to F_3 are now used in a second place, to filter error information. This process is called *weight transport*. In particular, all the weights w_{jk} in pathways fanning *out* from the jth F_2 node are transported for multiplication by the corresponding error terms δ_k; and the sum of all these products, times the bounding derivative term $f'(x_j)$, is back-coupled to adjust all the weights w_{ij} in pathways fanning *in* to the jth F_2 node.

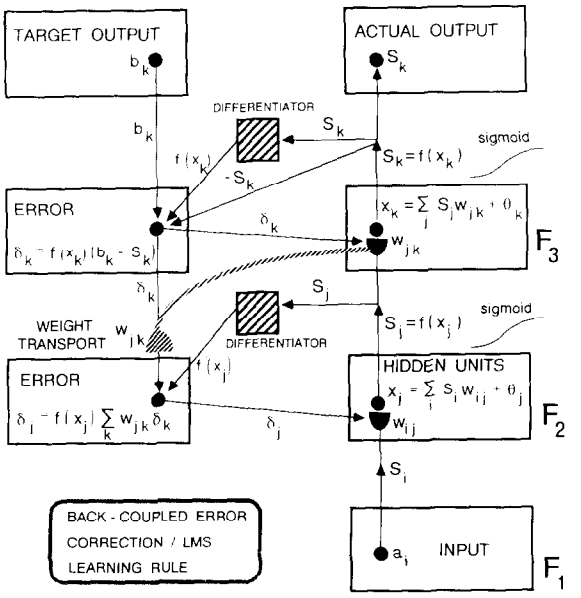


FIGURE 6. Block diagram of a back propagation algorithm for associative memory. Weights in the three-level feedforward perceptron are adjusted according to back-coupled error correction rules. Weight transport propagates error information in F_2-to-F_3 pathways back to weights in F_1-to-F_2 pathways.

9. HEBBIAN LEARNING

This brings us close to the present in this particular line of perceptron research. I am now going to step back and trace another major neural network theme that goes under the name *Hebbian learning*. One sentence in a 1949 book, *The Organization of Behavior*, by Donald Hebb is responsible for the phrase Hebbian learning:

> When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes place in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased. (Hebb, 1949)

Actually, "Hebbian learning" was not a new idea in 1949: it can be traced back to Pavlov and earlier. But in the decade of McCulloch and Pitts, the formulation of the idea in the above sentence crystallized the notion in such a way that it became widely influential in the emerging neural network field. Translated into a differential equation (Figure 7), the Hebbian rule computes a correlation between the presynaptic signal S_i and the postsynaptic activity x_j, with positive values of the correlation term $S_i x_j$ leading to increases in the weight w_{ij}.

The Hebbian learning theme has since evolved in a number of directions. One important development entailed simply adding a passive decay term to the

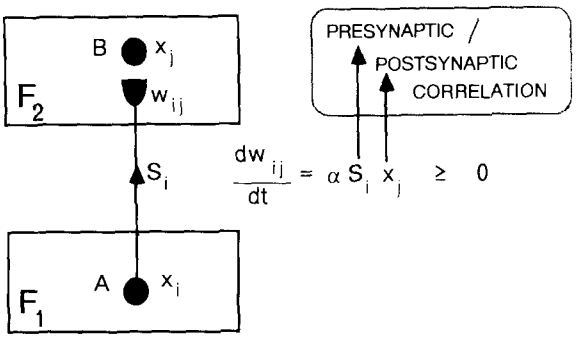


FIGURE 7. Donald Hebb (1949) provided a qualitative description of increases in path strength that occur when cell A helps to fire cell B. In the adaptive filter formalism, this hypothesis is often interpreted as a weight change that occurs when a presynaptic signal S_i is correlated with a postsynaptic activity x_j.

Hebbian correlation term:

$$\frac{dw_{ij}}{dt} = \alpha S_i x_j - w_{ij} \tag{6}$$

(Grossberg, 1968). Other developments are described below. In all these rules, changes in the weight w_{ij} depend upon a simple function of the presynaptic signal S_i, the postsynaptic activity x_j, and the weight itself, as in (6). In contrast, back-coupled error correction requires a term that must be computed away from the target node and then transmitted back to adjust the weight.

10. THE LEARNING MATRIX

Many of the models that followed the perceptron in the 1950s and 1960s can be phrased in Hebbian (plus McCulloch-Pitts) language. One of the earliest and most important is the learning matrix (Figure 8) developed by K. Steinbuch (1961). The function of the learning matrix is to sort, or partition, a set of vector patterns into categories. In the simple learning matrix illustrated in Figure 8a, an input pattern **a** is represented in the vertical wires. During learning a category for **a** is represented in the horizontal wires of the crossbar: **a** is placed in category J when the Jth component of the output vector **b** is set equal to 1. During such an input presentation the weight w_{iJ} is adjusted upward by a fixed amount if $a_i = 1$ and downward by the same amount if $a_i = 0$. Then during performance the weights w_{ij} are held constant; and an input **a** is deemed to be in category J if the weight vector $\mathbf{w}_J = (w_{1J}, \ldots w_{NJ})$ is closer than any other weight vector to **a**, according to some measure of distance.

Recasting the crossbar learning matrix in the adaptive filter format (Figure 8b) helps us to see that this simple model is the precursor of a fundamental module widely used in present day neural network modeling, namely *competitive learning*. In particular, activity at the top level of the learning matrix corresponds to a category representation. Setting activity x_J equal to 1, while all other x_j's are set equal to 0, corresponds to the dynamics of a *choice*, or *winner-take-all*, neural network. Steinbuch's learning rule can also be translated into the Hebbian formalism, with weight adjustment during learning a joint function of a presynaptic signal $S_i = (2a_i - 1)$ and a postsynaptic signal $x_j = b_j$. (This rule is not strictly Hebbian since weights can decrease as well as increase.) Then during performance, weight changes are prevented; a new signal function $S_i = a_i$ is chosen; and an F_2 choice rule is imposed, based, for example, on the dot product measure illustrated in Figure 9b.

A model comparative analysis of the learning matrix and the madaline models and their electronic implementations can be found in a paper by Steinbuch and Widrow (1965). This paper, entitled "A critical comparison of two kinds of adaptive classification networks," carries out a side-by-side analysis of the learning matrix and the madaline, tracing the two models' capabilities, similarities, and differences.

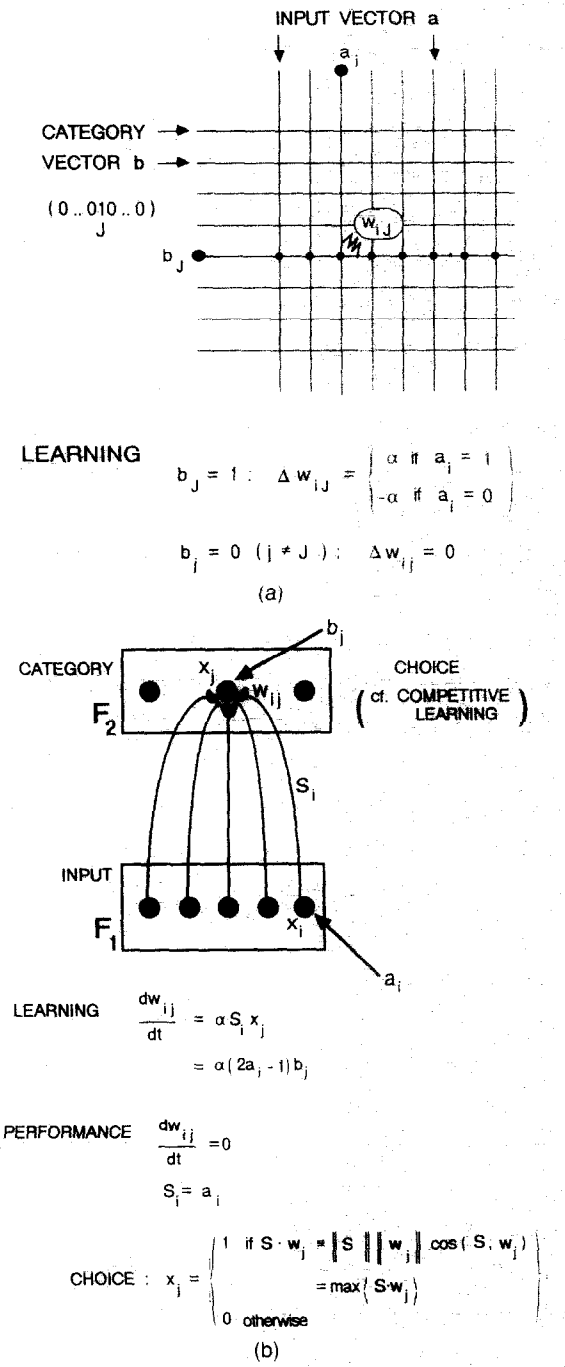


FIGURE 8. The learning matrix, for category learning. (a) Cross-bar architecture for electronic implementation. (b) The learning matrix in adaptive filter notation. The learning matrix was a precursor of the competitive learning paradigm.

11. LINEAR ASSOCIATIVE MEMORY (LAM)

We will now move to a different line of research, namely the linear associative memory (LAM)

models. Pioneering work on these models was done by Anderson (1972), Kohonen (1972), and Nakano (1972). Subsequently, many other linear associative memory models were developed and analyzed, for example, by Kohonen and his collaborators, who studied LAMs with iteratively computed weights that converge to the Moore-Penrose pseudoinverse (Kohonen & Ruohonen, 1973). This latter system is optimal with respect to the LMS error (5), and so is known as the optimal linear associative memory (OLAM) model. Variations included networks with partial connectivity, probabilistic learning laws, and nonlinear perturbations.

At the heart of all these variations is a very simple idea, namely that a set of pattern pairs $(\mathbf{a}^{(p)}, \mathbf{b}^{(p)})$ can be stored as a correlation weight matrix:

$$w_{ij} = \sum_{\substack{p \\ (\text{all patterns})}} a_i^{(p)} b_j^{(p)}. \tag{7}$$

The LAMs have been an enduringly useful class of models because, in addition to their great simplicity, they embody a sort of perfection. Namely, perfect recall is achieved, provided the input vectors $\mathbf{a}^{(p)}$ are mutually orthogonal. In this case, during performance, presentation of the pattern $\mathbf{a}^{(p)}$ yields an output vector $\mathbf{x}$ proportional to $\mathbf{b}^{(p)}$, as follows:

$$x_j \equiv \mathbf{a}^{(p)} \cdot \mathbf{w}_j = \sum_i a_i^{(p)} w_{ij} = \sum_i a_i^{(p)} \left(\sum_q a_i^{(q)} b_j^{(q)} \right)$$

$$= \sum_q \left(\sum_i a_i^{(p)} a_i^{(q)} \right) b_j^{(q)} = \sum_q (\mathbf{a}^{(p)} \cdot \mathbf{a}^{(q)}) b_j^{(q)}. \tag{8}$$

If, then, the vectors $\mathbf{a}^{(p)}$ are mutually orthogonal, the last sum in (8) reduces to a single term, with

$$x_j = \|\mathbf{a}^{(p)}\|^2 b_j^{(p)}. \tag{9}$$

Thus the output vector $\mathbf{x}$ is directly proportional to the desired output vector, $\mathbf{b}^{(p)}$. Finally, if we once again cast the LAM in the adaptive filter framework, we see that it is a Hebbian learning model (Figure 9).

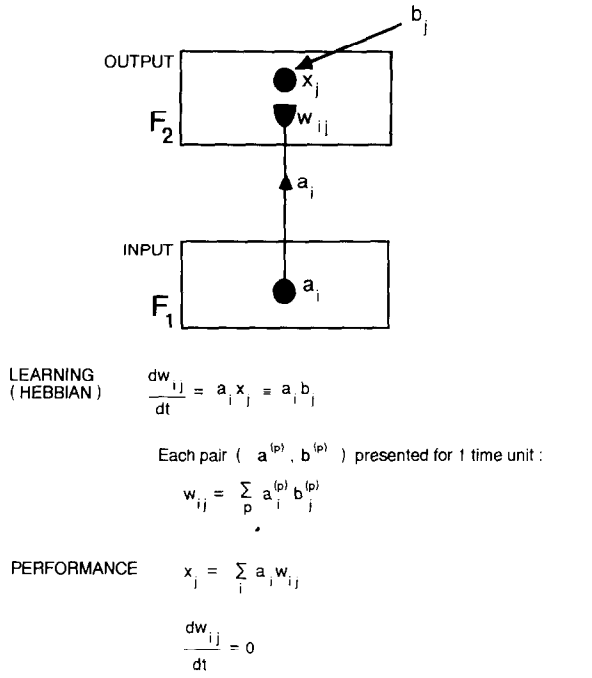


FIGURE 9. A linear associative memory network, in adaptive filter/Hebbian learning format.

12. REAL-TIME MODELS AND EMBEDDING FIELDS

Most of the models we have so far discussed require external control of system dynamics. In the back propagation model shown in Figure 6, for example, the initial feedforward activation of the three-level perceptron is followed by error correction steps that require either weight transport or reversing the direction of flow of activation. In the linear associative memory model in Figure 9, dynamics are altered as the system moves from its learning mode to its performance mode. During learning, activity x_j at the output level F_2 is set equal to the desired output b_j, while the input $\Sigma_i a_i w_{ij}$ coming to that level from F_1 through the adaptive filter is suppressed. During performance, in contrast, the dynamics are reversed: weight changes are supressed and the adaptive filter input determines x_j.

The phrase *real-time* describes neural network models that require no external control of system dynamics. (*Real-time* is alternatively used to describe any system that is able to process inputs as fast as they arrive.) Differential equations constitute the language of real-time models. A real-time model may or may not have an external teaching input, like the vector $\mathbf{b}$ of the LAM model; and learning may or may not be shut down after a finite time interval. A typical real-time model is illustrated in Figure 10. There, excitatory and inhibitory inputs could be either internal or external to the model, but, if present, the influence of a signal is not selectively ignored. Moreover the learning rate $\varepsilon(t)$ might, say, be constant or decay to 0 through time, but does not require algorithmic control. The dynamics of performance are described by the same set of equations as the dynamics of learning.

Real-time modeling has characterized the work of Stephen Grossberg over the past thirty years, work that in its early stages was called a theory of *embedding fields* (Grossberg, 1964). These early real-time models, as well as the more recent systems developed by Grossberg and his colleagues at the Boston University Center for Adaptive Systems, portray the inextricable linking of fast nodal activation and slow weight adaptation. There is no externally imposed

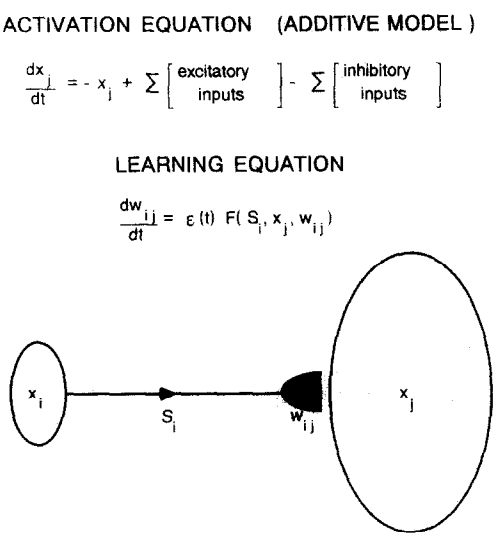


FIGURE 10. Elements of a typical real-time model, with additive activation equations.

distinction between a learning mode and a performance mode.

13. INSTARS AND OUTSTARS

Two key components of embedding field systems are the *instar* and the *outstar*. Figure 11 illustrates the fan-in geometry of the instar and the fan-out geometry of the outstar.

Instars often appear in systems designed to carry out adaptive coding, or content-addressable memory (CAM) (Kohonen, 1980). For example, suppose that the incoming weight vector $(w_{1J}, \ldots, w_{NJ})$ approaches the incoming signal vector $(S_1, \ldots, S_N)$ while an input vector **a** is present at F_1; and that the weight and signal vectors are normalized. Then eqn (2) implies that the filtered input $\Sigma_i S_i w_{iJ}$ to the Jth F_2 node approaches its maximum value during learning. Subsequent presentation of the same F_1 input pattern **a** maximally activates the Jth F_2 node; that is, the "content addresses the memory," all other things being equal.

The outstar, which is dual to the instar, carries out spatial pattern learning. For example, suppose that the outgoing weight vector $(w_{J1}, \ldots, w_{JN})$ approaches the F_1 spatial activity pattern $(x_1, \ldots, x_N)$ while an input vector **a** is present. Then subsequent activation of the Jth F_2 node transmits to F_1 the signal pattern $(S_J w_{J1}, \ldots, S_J w_{JN}) = S_J(w_{J1}, \ldots, w_{JN})$, which is directly proportional to the prior F_1 spatial activity pattern $(x_1, \ldots, x_N)$, even though the input vector is now absent; that is, the "memory addresses the content."

The upper instar and outstar in Figure 11 are examples of *heteroassociative* memories, where the field F_1 of nodes indexed by i is disjoint from the field F_2 of nodes indexed by j. In general, these fields

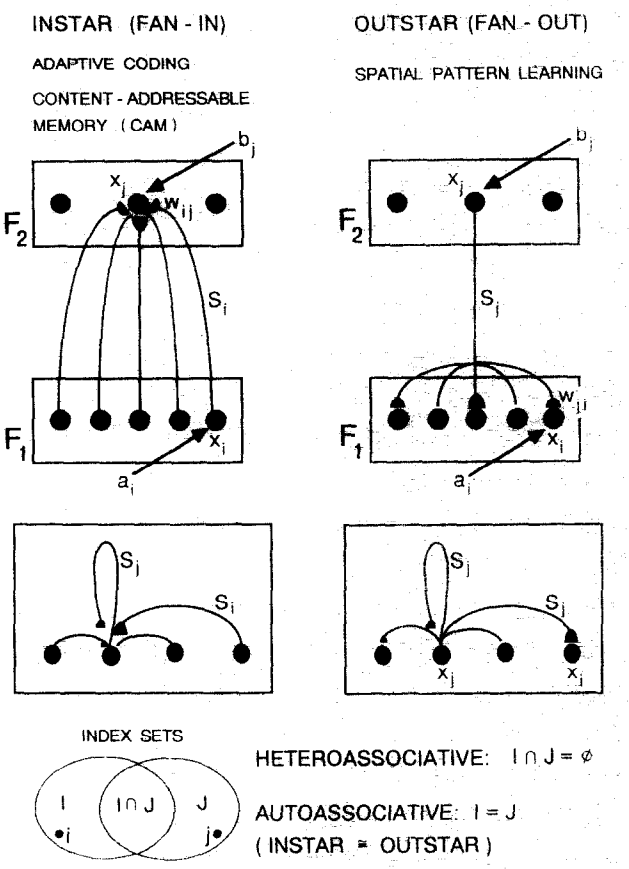


FIGURE 11. Heteroassociative and autoassociative instars and outstars, for adaptive coding and spatial pattern learning.

can overlap. The important special case in which the two fields coincide is called *autoassociative* memory, also shown in Figure 11. Powerful computational properties arise when neural network architectures are constructed from a combination of instars and outstars. We will later see some of these designs.

14. ADDITIVE AND SHUNTING ACTIVATION EQUATIONS

The outstar and the instar have been studied in great detail and with various combinations of activation, or short-term memory, equations and learning, or long-term memory, equations. One activation equation, the *additive model*, is illustrated in Figure 10. There, activity at a node is proportional to the difference between the net excitatory input and the net inhibitory input. Most of the models discussed so far employ a version of the additive activation model. For example, the McCulloch-Pitts activation equation (3) is the steady-state of the additive equation (10):

$$\frac{dx_j}{dt} = -x_j + \left[\sum_i S_i w_{ij} + \theta_j \right] - [\text{inhibition}]. \quad (10)$$

Grossberg (1988) reviews a number of neural models that are versions of the additive equation.

An important generalization of the additive model is the *shunting* model. In a shunting network, excitatory inputs drive activity toward a finite maximum, while inhibitory inputs drive activity toward a finite minimum, as in eqn (11):

$$\frac{dx_i}{dt} = -x_i + (A - x_i)\sum\begin{bmatrix}\text{excitatory}\\ \text{inputs}\end{bmatrix} - (B + x_i)\sum\begin{bmatrix}\text{inhibitory}\\ \text{inputs}\end{bmatrix}. \quad (11)$$

In (11), activity x_i remains in the bounded range $(-B, A)$, and decays to the resting level 0 in the absence of all inputs. In addition, shunting equations display other crucial properties such as normalization and automatic gain control. Finally, shunting network equations mirror the underlying physiology of single nerve cell dynamics, as summarized by the Hodgkin-Huxley (1952) equations:

$$\frac{dV}{dt} = -V + (V_{Na} - V) \times \bar{g}_{Na}m^3h - (V_K + V)\bar{g}_K n^4. \quad (12)$$

In this single nerve cell model, during depolarization, sodium ions entering across the membrane drive the potential V toward the sodium equilibrium potential V_{Na}; during repolarization, exiting potassium ions drive the potential toward the potassium equilibrium potential $-V_K$; and in the balance the cell is restored to its resting potential, which is here set equal to 0. In 1963 Hodgkin and Huxley won the Nobel Prize for their development of this classic neural model.

15. LEARNING EQUATIONS

A wide variety of learning laws for instars and outstars have also been studied. One example is the Hebbian correlation + passive decay equation (6). There, the weight w_{ij} computes a long-term weighted average of the product of presynaptic activity S_i and postsynaptic activity x_j.

A typical learning law for instar coding is given by eqn (13):

$$\frac{dw_{ij}}{dt} = \varepsilon(t)[S_i - w_{ij}]x_j. \quad (13)$$

Suppose, for example, that the Jth F_2 node is to represent a given category. According to (13), the weight vector $(w_{1J}, \ldots, w_{NJ})$ converges to the signal vector $(S_1, \ldots, S_N)$ when the Jth node is active; but that weight vector remains unchanged when a different category representation is active. The term x_J thus buffers, or *gates*, the weights w_{iJ} against undesired changes, including memory loss due to passive decay. On the other hand, a typical learning law for outstar pattern learning is given by eqn (14):

$$\frac{dw_{ij}}{dt} = \varepsilon(t)[x_i - w_{ij}]S_j. \quad (14)$$

In (14), when the Jth F_2 node is active the weight vector $(w_{J1}, \ldots, w_{JN})$ converges to the F_1 activity pattern vector $(x_1, \ldots, x_N)$. Again, a gating term buffers weights against inappropriate changes. Note that the pair of learning laws described by (13) and (14) are non-Hebbian, and are also nonsymmetric. That is, w_{ij} is generally not equal to w_{ji}, unless the F_1 and F_2 signal vectors **S** are identical to the corresponding activity vectors **x**.

A series of theorems encompassing neural network pattern learning by systems employing a large class of these and other activation and learning laws was proved by Grossberg in the late 1960s and early 1970s. One set of results falls under the heading *outstar learning theorems*. One of the most general of these theorems is contained in an article entitled "Pattern learning by functional-differential neural networks with arbitrary path weights" (Grossberg, 1972a). This is reprinted in *Studies of Mind and Brain* (see Bibliography), which also contains articles that introduce and analyze additive and shunting equations (10) and (11); learning with passive and gated memory decay laws (6), (13), and (14); outstar and instar modules; and neural network architectures constructed from these elements.

16. LEARNING SPACE-TIME PATTERNS: THE AVALANCHE

While most of the neural network models discussed in this article are designed to learn spatial patterns, problems such as speech recognition and motor learning require an understanding of space-time patterns as well. An early neural network model, called the *avalanche*, is capable of learning and performing an arbitrary space-time pattern (Grossberg, 1969). In essence, an avalanche is a series of outstars (Figure 12). During learning, the outstar active at time t

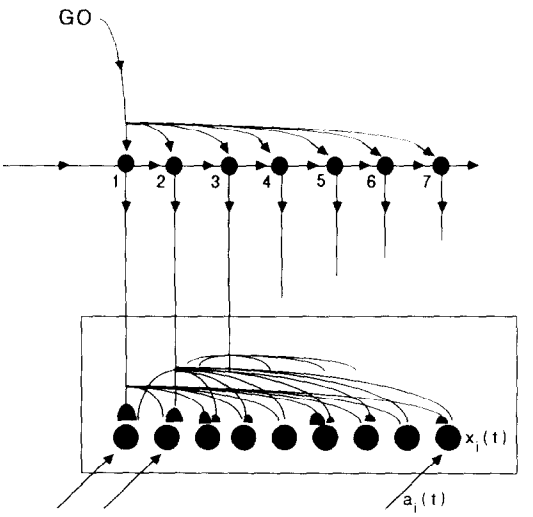


FIGURE 12. The avalanche: A neural network capable of learning and performing an arbitrary space-time pattern.

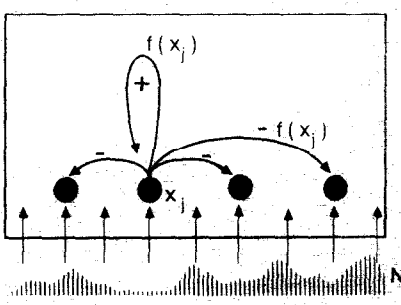
learns the spatial pattern $\mathbf{x}(t)$ generated by the input pattern vector $\mathbf{a}(t)$. It is useful to think of $\mathbf{x}(t)$ as the pattern determining finger positions for a piano piece: the same field of cells is used over and over, and the sequence ABC is not the same as CBA. Following learning, when no input patterns are present, activation of the sequence of outstars reads-out, or "performs," the space-time pattern it had previously learned. In its minimal form, this network can be realized as a single cell with many branches. Learning and performance can also be supervised by a nonspecific GO signal. The GO signal may terminate an action sequence at any time and otherwise modulate the performance energy and velocity. In general, the order of activation of the outstars, as well as the spatial patterns themselves, need to be learned. This can be accomplished using autoassociative networks, as in the theory of serial learning (Grossberg & Pepe, 1970) or adaptive signal processing (Hecht-Nielsen, 1981).

17. ADAPTIVE CODING AND CATEGORY FORMATION

Let us now return to the theme of adaptive coding and category formation, introduced earlier in our discussion of Steinbuch's learning matrix. As shown in Figure 8b, the learning matrix can be recast in the adaptive filter formalism, with the dynamics of the F_2 level defined in such a way that only one node is active at a given time. The active node, or category representation, is selected by a "teacher" during learning. During performance the active node is selected according to which weight vector forms the best match with the input vector. Now compare the learning matrix in Figure 8b with the instar in Figure 11. The pictures, or network "anatomies," seem to indicate that the instar is identical to the learning matrix. The difference between the two models lies in the dynamics, or network "physiology." The fundamental characteristic of the instar that distinguishes it from the learning matrix and other early models is the constraint that instar dynamics occur in real time. In particular, the instar filtered input $\mathbf{S} \cdot \mathbf{w}_j$ influences x_j at all times, and is not artificially suppressed during learning. However, the desire to construct a category learning system that can operate in real time immediately leads to many questions. The most pressing one is: how can the categories be represented if the dynamics are not imposed by an external agent? For the choice case, for example, the *internal* system dynamics need to allow at most one F_2 node to be active, even though other nodes may continue to receive large inputs, either internally, via the filter, or externally, via the vector **b**. Even when the category representation is a distributed pattern, this representation is generally a compressed, or contrast-enhanced, version of the highly distributed net pattern coming in to F_2 from all sources. This compression is, in fact, the step that carries out the process wherein some or many items are grouped into a new unit, or category.

18. SHUNTING COMPETITIVE NETWORKS

Fortunately, there is a well-defined class of neural networks ideally suited to play the role of the category representation field. This is the class of on-center/off-surround shunting competitive networks. Figure 13 illustrates one such system. There, the input vector **I** can be the sum of inputs from one or more sources and is, in general, highly distributed. *On-center* here refers to the feedback process whereby a cell sends net excitatory signals to itself and to its immediate neighbors; *off-surround* refers to the complementary process whereby the same cell sends net inhibitory signals to its more distant neighbors. In an article entitled "Contour enhancement, short-term memory, and constancies in reverberating neural networks," Grossberg (1973) carried out a mathematical characterization of the dynamics of various classes of shunting competitive networks. In particular he classified the systems according to the shape of the signal function $f(x_j)$. Depending upon whether this signal function is linear, faster-than-linear, slower-than-linear, or sigmoid, the networks are shown to quench or enhance low-amplitude noise; and to contrast-enhance or flatten the input pattern **I** in varying degrees. In particular, a faster-than-linear signal function implements the choice network needed for many models of category learning. A sigmoid signal function, on the other hand, suppresses noise and contrast-enhances the input pattern, without necessarily going to the extreme of concentrating all activity in one node. Thus an on-center/off-sur-



$$\frac{dx_j}{dt} = -x_j + (A - x_j)\left[I_j + f(x_j)\right] - x_j \sum_{k \neq j} f(x_k)$$

FIGURE 13. An on-center/off-surround shunting competitive network. Qualitative features of the signal function $f(x_j)$ determine the way in which the network transforms the input vector I into the state vector x.

round shunting competitive network with a sigmoid signal function is shown to be an ideal design for a category learning system with distributed code representations. This parametric analysis thus provided the foundation for constructing larger network architectures that use a competitive network as a component with well-defined functional properties.

19. COMPETITIVE LEARNING

A module of fundamental importance in recent neural network architectures is described by the phrase *competitive learning*. This module brings the properties of the learning matrix into the real-time setting. The basic competitive learning architecture consists of an instar filter, from a field F_1 to a field F_2, and a competitive neural network at F_2 (Figure 14). The competitive learning module can operate with or without an external teaching signal **b**; and learned changes in the adaptive filter can proceed indefinitely or cease after a finite time interval. If there is no teaching signal at a given time, then the net input vector to F_2 is the sum of signals arriving via the adaptive filter. Then if the category representation network is designed to make a choice, the node that automatically becomes active is the one whose weight vector best matches the signal vector, as in eqn (2). If there is a teaching signal, the category representation decision still depends on past learning, but this is balanced against the external signal **b**, which may or may not overrule the past in the competition. In either case, an instar learning law such as eqn (13) allows a chosen category to encode aspects of the new F_1 pattern in its learned representation.

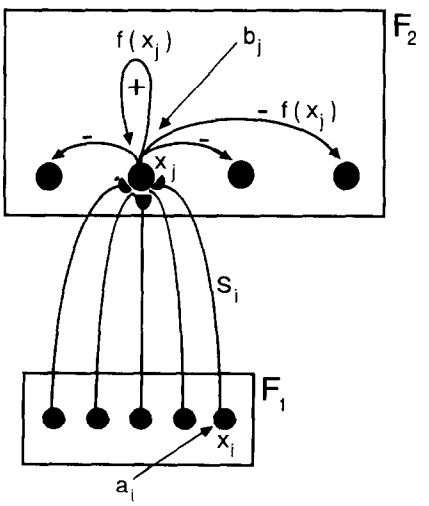


FIGURE 14. The basic competitive learning module combines the instar pattern coding system with a competitive network that contrast-enhances its filtered input.

20. COMPUTATIONAL MAPS

Investigators who have developed and analyzed the competitive learning paradigm over the years include Steinbuch (1961), Grossberg (1972b, 1976a, 1976b), von der Malsburg (1973), Amari (1977), Amari and Takeuchi (1978), Bienenstock, Cooper, and Munro (1982), Rumelhart and Zipser (1985), and many others. Moreover, these and other investigators proceeded to embed the competitive learning module in higher order neural network systems. In particular, systems were designed to learn computational maps, producing an output vector **b** in response to an input vector **a**. The core of many of these computational map models is in instar-outstar system. Recognition of this common theme highlights the models' differences as well as their similarities. An early self-organizing three-level instar-outstar computational map model was described by Grossberg (1972b), who later replaced the instar portion of this model with a competitive learning module (Grossberg, 1976b). The self-organizing feature map (Kohonen, 1984) and the counter-propagation network (Hecht-Nielsen, 1987) are also examples of instar-outstar competitive learning models.

The basic instar-outstar computational map system is depicted in Figure 15. The first two levels, F_1

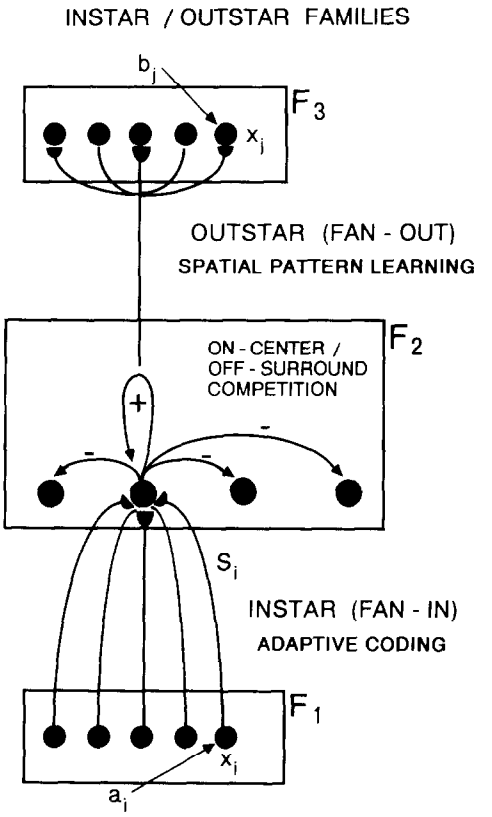


FIGURE 15. A three-level, feedforward instar-outstar module for computational mapping. The competitive learning module (F_1 and F_2) is joined with an outstar-type fan-out, for spatial pattern learning.

and F_2, form a competitive learning system. Included are the fan-in adaptive filter, contrast-enhancement at the "hidden" level F_2, and a learning law for instar coding of the input patterns **a**. The top two levels then employ a fan-out adaptive filter for outstar pattern learning of the vector **b**. This three-level architecture allows, for example, two very different input patterns to map to the same output pattern: each input pattern can activate its own compressed representation at F_2, while each of these F_2 representations can learn a common output vector. In the extreme case where each input vector **a** activates its own F_2 node the system learns any desired output. The generality of this extreme case, which implements an arbitrary mapping from $\mathbf{R}^m$ to $\mathbf{R}^n$, is offset by its lack of generalization, or continuity, as well as by the fact that each learned pair (**a**, **b**) requires its own F_2 node. Distributed F_2 representations provide greater generalization and efficiency, at a cost in complete a priori generality of the mapping.

21. INSTABILITY OF COMPUTATIONAL MAPS

The widespread use of instar-outstar families of computational maps attests to the power of this basic neural network architecture. This power is, however, diminished by the instability of feedforward systems: in general, recently learned patterns tend to erode past learning. This instability arises from two sources. First, even if a chosen category is the best match for a given input, that match may nevertheless be a poor one, chosen only because all the others are even worse. Established codes are thus vulnerable to recoding by "outliers." Second, learning laws such as eqn (13) imply that a weight vector tends toward a new vector that encodes the presently active pattern, thereby weakening the trace of the past. Thus weight vectors can eventually drift far from their original patterns, even if learning is very slow and even if each individual input makes a good match with the past as recorded in the weights.

The many existing variations on the three-level instar-outstar theme illustrate some of the ways in which this family of models can be adapted to cope with the basic system's intrinsic instability. One stabilization technique causes learning to slow or cease after an initial finite interval; but then a subsequent unexpected pattern cannot be encoded, and instability could still creep in during the initial learning phase. Another approach is to restrict the class of input patterns to a stable set. This technique requires that the system can be sufficiently well analyzed to identify such a class, like the orthogonal inputs of the linear associative memory model (Figure 9); and that all inputs can be confined to this class. An often successful way to compensate for the instability of these systems is to slow the learning rate to such an extent that learned patterns are buffered against massive recoding by any single input. Of course, then, each pattern needs to be presented very many times for adequate learning to occur, a fact that was discussed, for example, by Rosenblatt in his critique of back propagation.

22. ADAPTIVE RESONANCE THEORY (ART)

It was analysis of the instability of feedforward instar-outstar systems that led to the introduction of adaptive resonance theory (ART) (Grossberg, 1976c) and to the development of the neural network systems ART 1 and ART 2 (Carpenter & Grossberg, 1987a, 1987b). ART networks are designed, in particular, to resolve the *stability-plasticity dilemma*: they are stable enough to preserve significant past learning, but nevertheless remain adaptable enough to incorporate new information whenever it might appear.

The key idea of adaptive resonance theory is that the stability-plasticity dilemma can be resolved by a system in which the three-level network of Figure 15 is folded back on itself, identifying the top level (F_3) with the bottom level (F_1) of the instar-outstar mapping system. Thus the minimal ART module includes a bottom-up competitive learning system combined with a top-down outstar pattern learning system. When an input **a** is presented to an ART network, system dynamics initially follow the course of competitive learning (Figure 14), with bottom-up activation leading to a contrast-enhanced category representation at F_2. In the absence of other inputs to F_2, the active category is determined by past learning as encoded in the adaptive weights in the bottom-up filter. But now, in contrast to feedforward systems, signals are sent from F_2 back down to F_1 via a top-down adaptive filter. This feedback process allows the ART module to overcome both of the sources of instability described in section 21, as follows.

First, as in the competitive learning module, the category active at F_2 may poorly match the pattern active at F_1. The ART system is designed to carry out a matching process that asks the question: should this input really be in this category? If the answer is no, the selected category is quickly rendered inactive, before past learning is disrupted by the outlier, and a search process ensues. This search process employs an auxiliary *orienting subsystem* that is controlled by the dynamics of the ART system itself. The orienting subsystem incorporates a dimensionless *vigilance parameter* that establishes the criterion for deciding whether the match is a good enough one for the input to be accepted as an exemplar of the chosen category.

Second, once an input is accepted and learning proceeds, the top-down filter continues to play a different kind of stabilizing role. Namely, top-down signals that represent the past learning meet the original input signals at F_1. Thus the F_1 activity pattern is a function of the past as well as the present, and it is this blend of the two, rather than the present input alone, that is learned by the weights in both adaptive filters. This dynamic matching during learning leads to stable coding, even with fast learning.

An example of the ART 1 class of minimal modules is illustrated in Figure 16. In addition to the two adaptive filters and the orienting subsystem, Figure 16 depicts gain control processes that actively regulate learning. Theorems have been proved to characterize the response of an ART 1 module to an arbitrary sequence of binary input patterns (Carpenter & Grossberg, 1987a). ART 2 systems were developed to self-organize recognition categories for analog as well as binary input sequences. One principal difference between the ART 1 and the ART 2 modules is shown in Figure 17. In examples so far developed, the stability criterion for analog inputs has required a three-layer feedback system within the F_1 level: a bottom layer where input patterns are read in; a top layer where filtered inputs from F_2 are read in; and a middle layer where the top and bottom patterns are brought together to form a matched pattern that is then fed back to the top and bottom F_1 layers.

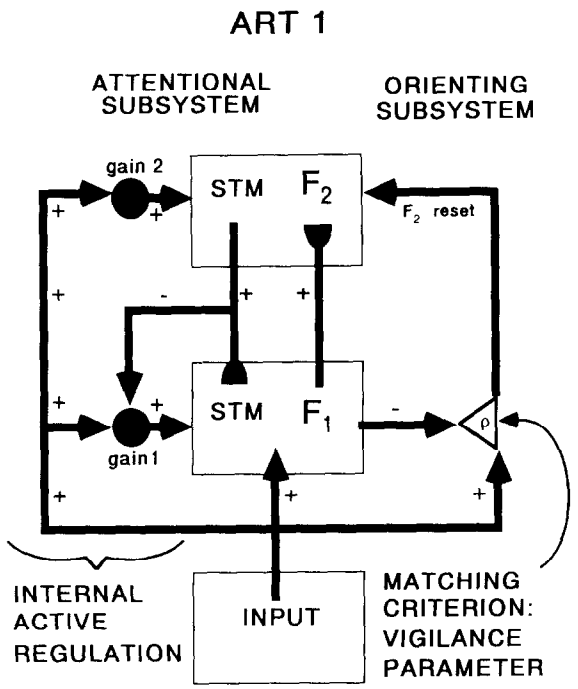


FIGURE 16. An ART 1 module for stable, self-organizing categorization of an arbitrary sequence of binary input patterns.

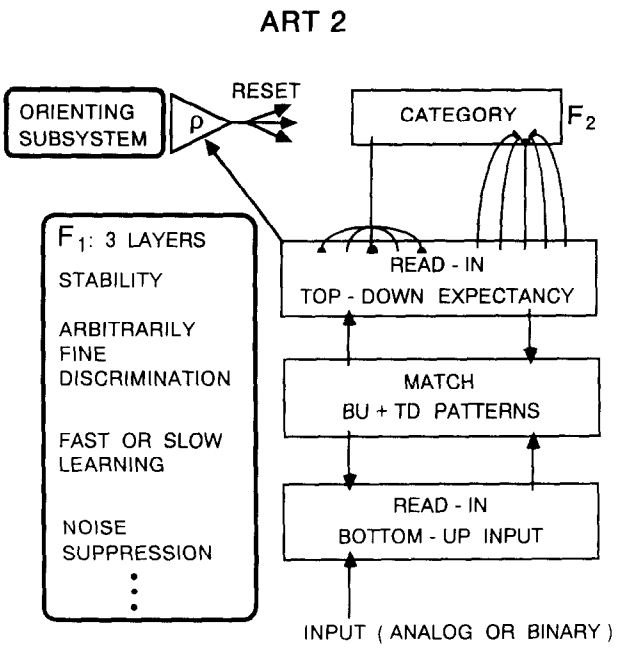


FIGURE 17. Principal elements of an ART 2 module for stable, self-organizing categorization of an arbitrary sequence of analog or binary input patterns. The F_1 level is a competitive network with three processing layers.

23. ART FOR ASSOCIATIVE MEMORY

A minimal ART module is a category learning system that self-organizes a sequence of input patterns into various recognition categories. It is not an associative memory system. However, like the competitive learning module in the 1970s, a minimal ART module can be embedded in a larger system for associative memory. A system such as an instar-outstar module (Figure 15) or a back-propagation algorithm (Figure 6) directly pairs sequences of individual *vectors* (**a**, **b**) during learning. If an ART system replaces levels F_1 and F_2 of the instar-outstar module, the associative learning system becomes self-stabilizing. ART systems can also be used to pair sequences of the *categories* self-organized by the input sequences (Figure 18). Moreover, the symmetry of the architecture implies that pattern recall can occur in either direction during performance. This scheme brings to the associative memory paradigm the code compression capabilities of the ART system, as well as its stability properties.

24. COGNITRON AND NEOCOGNITRON

In conclusion, we will consider two sets of models that are variations on the themes previously described. The first class, developed by Kunihiko Fukushima, consists of the cognitron (Fukushima, 1975) and the larger-scale neocognitron (Fukushima, 1980, 1988). This class of neural models is distin-

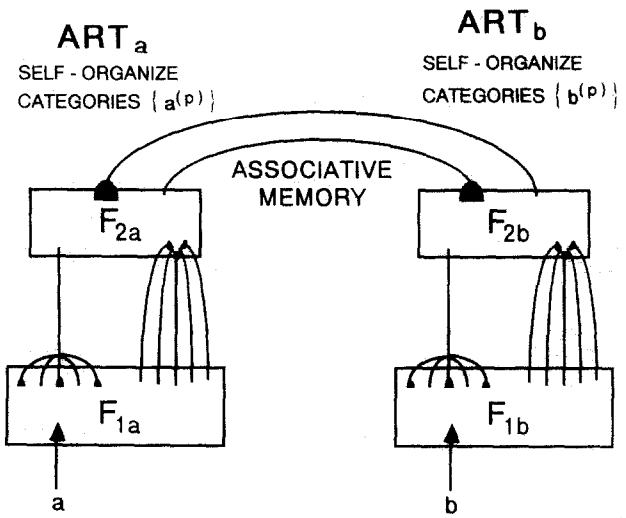


FIGURE 18. Two ART systems combined to form an associative memory architecture.

guished by its capacity to carry out translation-invariant and size-invariant pattern recognition. This is accomplished by redundantly coding elementary features in various positions at one level; then cascading groups of features to the next level; then groups of these groups; and so on. Learning can proceed with or without a teacher. Locally the computations are a type of competitive learning that use combinations of additive and shunting dynamics.

25. SIMULATED ANNEALING

Finally, in addition to the probabilistic weight change laws which were a prominent feature of, for example, the modeling efforts of pioneers such as Rosenblatt and Amari, another class of probabilistic weight change laws appears in more recent work under the name *simulated annealing*, introduced by Kirkpatrick, Gellatt, and Vecchi (1983). The main idea of simulated annealing is the transposition of a method from statistical mechanics, namely the Metropolis algorithm (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953), into the general context of large complex systems. The Metropolis algorithm provides an approximate description of a many-body system, namely a material that anneals into a solid as temperature is slowly decreased. Kirkpatrick et al. (1983) drew an analogy between this system and problems of combinatorial optimization, such as the traveling salesman problem, where the goal is to minimize a cost function. The methods and ideas, as well as the large scale nature of the problem, are so closely tied to those of neural networks that the two approaches are often linked. This link is perhaps closest in the Boltzmann machine (Ackley, Hinton, & Sejnowski, 1985), which uses a simulated annealing algorithm to update weights in a binary network similar to the additive model studied by Hopfield (1982).

26. CONCLUSION

We have seen how the adaptive filter formalism is general enough to describe a wide variety of neural network modules for associative memory, category learning, and pattern recognition. Many systems developed and applied in recent years are variations on one or more of these modular themes. This approach can thus provide a core vocabulary and grammar for further analysis of the rich and varied literature of the neural network field.

REFERENCES

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, **9**, 147–169.

Amari, S.-I. (1972). Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on Computers*, **C-21**, 1197–1206.

Amari, S.-I. (1977). Neural theory of association and concept-formation. *Biological Cybernetics*, **26**, 175–185.

Amari, S.-I., & Takeuchi, A. (1978). Mathematical theory on formation of category detecting nerve cells. *Biological Cybernetics*, **29**, 127–136.

Anderson, J. A. (1972) A simple neural network generating an interactive memory. *Mathematical Biosciences*, **14**, 197–220.

Bienenstock, E., Cooper, L. N., & Munro, P. W. (1982). A theory for the development of neuron selectivity: Orientation specificity and binocular interaction in the visual cortex. *Journal of Neuroscience*, **2**, 32–48.

Carpenter, G. A., & Grossberg, S. (1987a). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, **37**, 54–115.

Carpenter, G. A., & Grossberg, S. (1987b). ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, **26**, 4919–4930.

Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, **20**, 121–136.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, **36**, 193–202.

Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, **1**, 119–130.

Grossberg, S. (1964). *The theory of embedding fields with applications to psychology and neurophysiology*. New York: Rockefeller Institute for Medical Research.

Grossberg, S. (1968). Some nonlinear networks capable of learning a spatial pattern of arbitrary complexity. *Proceedings of the National Academy of Sciences USA*, **59**, 368–372.

Grossberg, S. (1969). Some networks that can learn, remember, and reproduce any number of complicated space-time patterns. I. *Journal of Mathematics and Mechanics*, **19**, 53–91.

Grossberg, S. (1972a). Pattern learning by functional-differential neural networks with arbitrary path weights. In K. Schmitt (Ed.), *Delay and functional differential equations and their applications* (pp. 121–160). New York: Academic Press.

Grossberg, S. (1972b). Neural expectation: Cerebellar and retinal analogs of cells fired by learnable or unlearned pattern classes. *Kybernetik*, **10**, 49–57.

Grossberg, S. (1973). Contour enhancement, short-term memory, and constancies in reverberating neural networks. *Studies in Applied Mathematics*, **52**, 217–257.

Grossberg, S. (1976a). On the development of feature detectors

in the visual cortex with applications to learning and reaction-diffusion systems. *Biological Cybernetics*, **21**, 145–159.
Grossberg, S. (1976b). Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, **23**, 121–134.
Grossberg, S. (1976c). Adaptive pattern classification and universal recoding, II: Feedback, expectation, olfaction, and illusions. *Biological Cybernetics*, **23**, 187–202.
Grossberg, S. (1988). Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural Networks*, **1**, 17–61.
Grossberg, S., & Pepe, J. (1970). Schizophrenia: Possible dependence of associational span, bowing, and primacy vs. recency on spiking threshold. *Behavioral Science*, **15**, 359–362.
Hebb, D. O. (1949). *The organization of behavior*. New York: Wiley.
Hecht-Nielsen, R. (1981). Neural analog information processing. *Proceedings of the Society of Photo-Optical Instrumentation Engineers*, **298**, 138–141.
Hecht-Nielsen, R. (1987). Counterpropagation networks. *Applied Optics*, **26**, 4979–4984.
Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, **117**, 500–544.
Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences USA*, **79**, 2554–2558.
Kirkpatrick, S., Gelatt, C. D., Jr., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, **220**, 671–680.
Kohonen, T. (1972). Correlation matrix memories. *IEEE Transactions on Computers*, **C-21**, 353–359.
Kohonen, T. (1980). *Content-addressable memories*. Berlin: Springer-Verlag.
Kohonen, T. (1984). *Self-organization and associative memory*. Berlin: Springer-Verlag.
Kohonen, T., & Ruohonen, M. (1973). Representation of associated data by matrix operators. *IEEE Transactions on Computers*, **C-22**, 701–702.
Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, **21**, 1087–1091.
McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, **9**, 127–147.
Nakano, N. (1972). Associatron: A model of associative memory. *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-2**, 381–388.
Parker, D. (1982). *Learning logic*. Invention report, **S81-64**, File 1, Office of Technology Licensing, Stanford University.
Pitts, W., & McCulloch, W. S. (1947). How we know universals: The perception of auditory and visual forms. *Bulletin of Mathematical Biophysics*, **9**, 127–147.
Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**, 386–408.
Rosenblatt, R. (1962). *Principles of neurodynamics*. Washington, DC: Spartan Books.
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructures of cognitions*, **I** (pp. 318–362) Cambridge, MA: MIT Press.
Rumelhart, D. E., & Zipser, D. (1985). Feature discovery by competitive learning. *Cognitive Science*, **9**, 75–112.
Russell, B., & Whitehead, A. N. (1910, 1912, 1913) *Principia mathematica* I–III. Cambridge: Cambridge University Press.
Steinbuch, K. (1961) Die Lernmatrix. *Kybernetik*, **1**, 36–45.
Steinbuch, K., & Widrow, B. (1965). A critical comparison of two kinds of adaptive classification networks. *IEEE Transactions on Electronic Computers*, **EC-14**, 737–740.
von der Malsburg, C. (1973). Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, **14**, 85–100.
von Neumann, J. (1958). *The computer and the brain*. New Haven: Yale University Press.
Werbos, P. J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Unpublished Ph.D. thesis, Harvard University, Cambridge, MA.
Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, **1**, 339–356.
Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits. *1960 IRE WESCON Convention Record*, part 4, 96–104.
Widrow, B., & Winter, R. (1988). Neural nets for adaptive filtering and adaptive pattern recognition. *Computer*, **21**, 25–39.

BIBLIOGRAPHY

Additional selected readings can be found in the collections listed below.

A. Collections of Articles

Amari, S.-I., & Arbib, M. A. (Eds.). (1982). *Competition and cooperation in neural nets. Lecture Notes in Biomathematics*, **45**. Berlin: Springer-Verlag.
Anderson, J. A., & Rosenfeld, E. (Eds.). (1988). *Neurocomputing: Foundations of research*. Cambridge, MA: MIT Press.
Carpenter, G. A., & Grossberg, S. (1987). *Applied Optics*, **26**(23). [Special issue on Neural Networks].
Grossberg, S. (Ed.). (1981). *Mathematical psychology and psychophysiology*. Providence: American Mathematical Society.
Grossberg, S. (1982). *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*. Boston: Reidel/Kluwer.
Grossberg, S. (1988). *Neural networks and natural intelligence*. Cambridge, MA: MIT Press.
McCulloch, W. S. (Ed.). (1965). *Embodiments of mind*. Cambridge, MA: MIT Press.
Rumelhart, D., McClelland, J., and the PDP Research Group. (1986). *Parallel distributed processing*. Cambridge, MA: MIT Press.
Sanders, A. C., & Zeevi, Y. Y. (Eds.). (1983). *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-13**(5) [Special issue on Neural and Sensory Information Processing].
Shriver, B. (Ed.). (1988). *Computer*, **21**(3). [Special issue on Artificial Neural Systems].
Szu, H. H. (Ed.). (1987). *Optical and hybrid computing*. Bellingham, WA: The Society of Photo-Optical Instrumentation Engineers.

B. Journals

Kybernetik (1961–1974); *Biological Cybernetics* (1975–).
Neural Networks (1988–).

C. Reviews

Kohonen, T. (1987). Adaptive, associative, and self-organizing functions in neural computing. *Applied Optics*, **26**, 4910–4918.
Levine, D. (1983). Neural population modeling and psychology: A review. *Mathematical Biosciences*, **66**, 1–86.
Simpson, P. K. (in press). *Artificial neural systems: Foundations, paradigms, applications, and implementations*. Elmsford, NY: Pergamon Press.