Summarized CPN

1987

Same article as in Applied Optics

Counterpropagation Networks

Robert Hecht-Nielsen
Hecht-Nielsen Neurocomputer Corporation
5893 Oberlin Drive
San Diego, CA 92121
619-546-8877

Abstract

By combining Kohonen learning and Grossberg learning a new type of mapping neural network is obtained. This *counterpropagation* network (CPN) functions as a statistically optimal self-programming lookup table. The paper begins with some introductory comments, followed by the definition of the CPN network. Then a closed-form formula for the error of the network is developed. The paper concludes with a discussion of CPN variants and comments about CPN convergence and performance. A neurocomputing bibliography with 79 entries is provided.

1 Introduction

Of all of the practical information processing operations that neural networks can currently carry out, one of the most useful is the ability to learn a mathematical mapping by self-organization in response to examples of the mapping's action. Typically, one generates a set of examples $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots$ of the action of a function ϕ, where $\mathbf{y}_i = \phi(\mathbf{x}_i)$ or $\mathbf{y}_i = \phi(\mathbf{x}_i) + \mathbf{n}$ ($\mathbf{n}$ is a stationary noise process). These examples statistically define the desired input / output relationship.

A network that self-organizes itself to implement an approximation to a function or mapping is called a *mapping neural network*. Currently, the most popular mapping neural network is the backpropagation network of Rumelhart [64]. The backpropagation network has been shown to be capable of implementing approximations to a variety of mappings from $\mathbf{R}^n$ to $\mathbf{R}^m$. The approximation achieved has been shown to be optimal in a certain least mean squared error sense. Although most applications of backpropagation to date have been to mappings that have binary input and output vectors (i.e., each vector component is essentially either one or zero), there is ample evidence that backpropagation also works for at least some non- binary vector mappings.

In this paper a new type of mapping neural network, called the *counterpropagation network* (CPN), is introduced. It is shown that (under nonpathological conditions) this network will self-organize a near-optimal (in the sense that the entries in the "table" are statistically equiprobable) lookup table approximation to the mapping used to generate its data. The method works equally well for both binary and continuous vector mappings. It is shown that for a sufficiently large network the

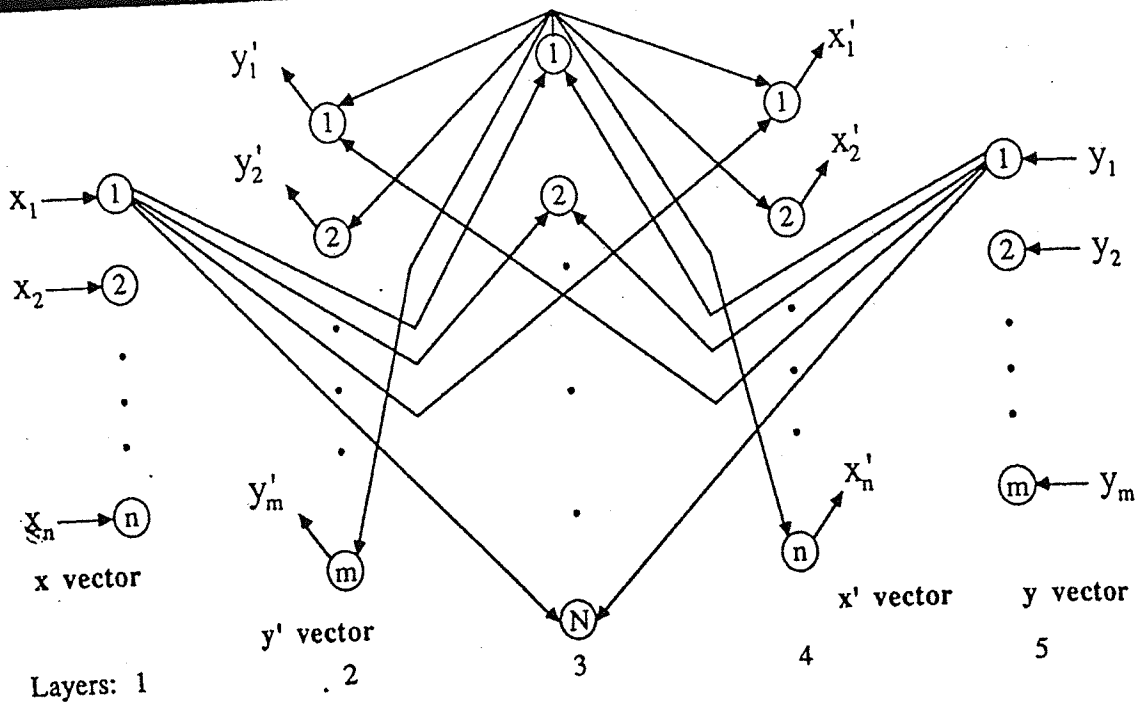

Figure 1: Topology of the Counterpropagation Network

mapping approximation can be made essentially as accurate as desired. The counterpropagation network architecture is a combination of a portion of the self-organizing map of Kohonen [51] and the outstar structure of Grossberg [31].

2 The Counterpropagation Network

Figure 1 presents the topology of the CPN and Figure 2 presents the transfer functions and learning laws of each of the 5 layers of the network. The basic idea is that, during adaptation, pairs of example vectors $(\mathbf{x}, \mathbf{y})$ (both assumed to be of unit length) are presented to the network at layers 1 and 5, respectively. These vectors then *propagate* through the network in a *counterflow* manner to yield output vectors $\mathbf{x}'$ and $\mathbf{y}'$ that are intended to be approximations of $\mathbf{x}$ and $\mathbf{y}$. Thus the name *counterpropagation*.

As shown in Figure 1, the full $\mathbf{x}$ and $\mathbf{y}$ input vectors are supplied to each processing element of layer 3, whereas the processing elements of layers 2 and 4 only get their corresponding component of these vectors. The processing elements of layers 2 and 4 also receive inputs from each processing element of layer 3 (the layer 3 processing element output signals, of which there are N, are labeled as z_i's).

When an $\mathbf{x}$ and $\mathbf{y}$ vector pair is presented to the network the processing elements of layer 3 compete with one another as shown in the z_i equation of Figure 2. The processing element with the highest average weight vector correlation with $\mathbf{x}$ and $\mathbf{y}$ (i.e., the highest value of I_i) has its output signal (z_i) set to one. All $N-1$ other z_i outputs are set to zero. Ties are broken on the basis of the

Layers 1 & 5: Fanouts

Layers 2 & 4: $y'_i = \sum_{j=1}^{n} w_{ij}\ z_j$

$$\dot{w}_{ij} = (-\ a\ w_{ij} + b\ y_i)\ z_j$$

(x'_i similar)

Layer 3: $z_i = \begin{cases} 1 \text{ if } I_i > I_j\ , \forall_j \\ 0 \text{ otherwise} \end{cases}$ $\qquad I_i = \sum_{j=1}^{n} u_{ij} x_j + \sum_{j=1}^{m} v_{ij}\ y_j$

$$= \mathbf{u}_i \cdot \mathbf{x} + \mathbf{v}_i \cdot \mathbf{y}$$

$$\dot{\mathbf{u}}_i = \alpha\ (\mathbf{x} - \mathbf{u}_i)\ z_i \qquad \dot{\mathbf{v}}_i = \beta\ (\mathbf{y} - \mathbf{v}_i)\ z_i$$

Figure 2: CPN Mathematics

smallest processing element index. By virtue of the form of the layer 3 weight change law (shown at the bottom of Figure 2), only the best matching processing element on layer 3 gets to adjust its weight vector in response to each vector pair input during training.

As shown by Kohonen [51], the $(\mathbf{u}_i, \mathbf{v}_i)$ weight vectors of layer 3 (which start out as unit vectors) will self-organize in response to the input vector pairs so that these vectors will, after statistical equilibration, have the following two properties:

1. The vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ will remain approximately unit vectors.
2. The weight vectors $(\mathbf{u}_i, \mathbf{v}_i)$ will be distributed on the cartesian product unit sphere in such a way that the probability of a given randomly chosen $(\mathbf{x}, \mathbf{y})$ (the x vectors are assumed to be chosen in accordance with a fixed probability density function ρ and this then induces a density for the y vectors) being closest to any one of these weight vectors is equal to roughly $1/N$. In other words, the win regions of these weight vectors are equally likely to contain an input vector pair chosen in accordance with ρ (see the next Section and [37] for the definition of a win region). [Note: Proposition 5.1 in Chapter 5 of [51] is not exactly correct (personal communication from T. Kohonen). A density correction factor is required because the weight vectors tend to oversample high density areas and undersample low density areas. A new edition of this book (with this correction) is due out in late 1987. A practical cure for this problem is suggested below.]

Thus, the weight vectors of the processing elements of layer 3 become organized as a more-or-less statistically optimal set of examples of the relationships between the x and y vectors. Note that if

the function ϕ is invertable that layer 3 will be equally sensitive to both the forward and backwards mappings. Weight changes within layer 3 processing elements take place in accordance with the two equations at the bottom of Figure 2. This *Kohonen learning law* rotates the closest matching weight vectors around towards the latest input vectors. This movement is typically almost perpendicular to the weight vector and thus has the effect of keeping the vector approximately normalized (see [51] for details on Kohonen learning).

Layers 2 and 4 of the network learn the average $\mathbf{x}$ and $\mathbf{y}$ vector values that occur when each of the processing elements of layer 3 wins the closeness competition. Such a structure is called an *outstar* and was invented by Grossberg [34,31]. Grossberg has shown that if the constants a and b in the layer 2 and 4 equations of Figure 2 are set equal to each other at a positive value less than one, that the $\mathbf{x}'$ and $\mathbf{y}'$ values emitted by layers 2 and 4 will, after statistical equilibrium, equal approximately the average value of the $\mathbf{x}$ and $\mathbf{y}$ vectors that historically allowed that particular layer 3 element to win the competition of that layer.

In summary, after both layer 3 and layers 2 and 4 equilibrate, an input vector pair $\mathbf{x}$ and $\mathbf{y}$ will cause the network to emit $\mathbf{x}'$ and $\mathbf{y}'$ vectors that are approximately the same as the nearest matching layer 3 weight vector pair. If the pair $\mathbf{x}$ and 0 (i.e., the $\mathbf{y}$ vector is set to zero) are entered, then the output pair will again be essentially the same as the layer 3 weight vector pair with a first vector element that best matches $\mathbf{x}$. If ϕ is invertable the same sort of result will hold if a pair 0 and $\mathbf{y}$ is entered. If a pair having some components of both the $\mathbf{x}$ and $\mathbf{y}$ vectors zeroed out is entered, the network will *complete* the vector pair and the outputs will be approximately the same as the best matching layer three weight vector's elements. Thus, CPN functions very much like a lookup table. The overall CPN network can be viewed as a modular element that will self-organize to form such a table and then respond to partial inputs as outlined above. Figure 3 shows a schematic representation for this CPN module. As discussed below, this module can be used as a building block for creating more complex and more capable networks.

3 CPN Error Analysis

Since the output of a CPN is almost exactly equal to one of the weight vectors (this is also true for the "forward only" CPN version considered below), let us analyze the mean squared error of a CPN system on this basis. In order to stay within the confines of conventional notation (and to emphasize the universality of the discussion) the layer 3 weight vectors shall be named $\mathbf{w}_i$'s, even though this contradicts the notation of the last Section. In fact, several of the variables (e.g., $n, \mathbf{z}, \mathbf{v}$, etc.) used previously will be reused in this Section with different meanings.

The first issue is to define the input/output behavior of the CPN system. The basic idea is that an input vector $\mathbf{z}$ (consisting perhaps of the input vector pair $\mathbf{x}$ and $\mathbf{y}$ of the last Section) comes into the system. The network then compares $\mathbf{z}$ with all of the weight vectors $\mathbf{w}_i$, $i = 1, 2, \ldots, N$ of the system and emits the nearest matching vector $\mathbf{w}_k$ as its output. This is the basic function of the CPN network. As in the previous section, $\mathbf{z}$ is assumed to be randomly selected in accordance with a fixed probability density function ρ.

Given the above definitions, the mean squared error $F(W)$ of this system is the average of the

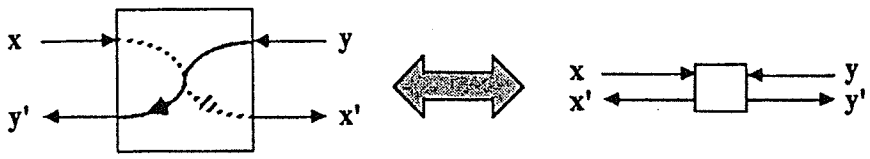

Figure 3: The CPN Module

square of the distance between an arbitrary input vector $\mathbf{z}$ and the nearest matching $\mathbf{w}_i$, weighted by the probability $\rho(\mathbf{z})$; which is given by

$$F(W) = \int_{\Omega^n} \theta(W, \mathbf{z})\, \rho(\mathbf{z})\, dA(\mathbf{z})$$

where,

$$W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N]$$

$$\mathbf{z},\ \mathbf{w}_i \in \Omega^n \equiv \{\mathbf{x} \in \mathbf{R}^n \,||\mathbf{x}| = 1\} \subset \mathbf{R}^n$$

$$\theta(W, \mathbf{z}) \equiv MIN(|\mathbf{z} - \mathbf{w}_1|^2, |\mathbf{z} - \mathbf{w}_2|^2, \ldots, |\mathbf{z} - \mathbf{w}_N|^2)$$

This can be simplified by noting that

$$|\mathbf{z} - \mathbf{w}_i|^2 = |\mathbf{z}|^2 + |\mathbf{w}_i|^2 - 2\,\mathbf{z} \cdot \mathbf{w}_i$$

and thus

$$\theta(W,\mathbf{z}) = 2 - 2\,MAX(\mathbf{z}\cdot\mathbf{w}_1, \mathbf{z}\cdot\mathbf{w}_2, \ldots, \mathbf{z}\cdot\mathbf{w}_N)$$

By defining the *win region* of $\mathbf{w}_i$ to be

$$B_i \equiv \{\mathbf{z} \in \Omega^n | i \text{is the smallest integer } 1 \le i \le N \text{such that } \mathbf{z}\cdot\mathbf{w}_i \ge \mathbf{z}\cdot\mathbf{w}_j \ \forall j\}$$

we can reexpress the integral for $F(W)$ as

$$F(W) = 2 - 2\sum_{i=1}^{N} \mathbf{w}_i \cdot \int_{B_i} \mathbf{z}\,\rho(\mathbf{z})\,dA(\mathbf{z})$$

since for $\mathbf{z} \in B_i$

$$\mathbf{z}\cdot\mathbf{w}_i = MAX(\mathbf{z}\cdot\mathbf{w}_1, \mathbf{z}\cdot\mathbf{w}_2, \ldots, \mathbf{z}\cdot\mathbf{w}_N)$$

and since $\mathbf{w}_i$ is a constant vector. Note that if m_i is defined to be

$$m_i \equiv \int_{B_i} \rho(\mathbf{z}) dA(\mathbf{z})$$

and $\mathbf{v}_i$ (the centroid of the sphere subset B_i) is defined to be

$$\mathbf{v}_i \equiv (1/m_i) \int_{B_i} \mathbf{z}\,\rho(\mathbf{z})\,dA(\mathbf{z})$$

then we can rewrite $F(W)$ as

$$F(W) = 2 - 2\sum_{i=1}^{N} m_i\,(\mathbf{w}_i\cdot\mathbf{v}_i).$$

Note that

$$\sum_{i=1}^{N} m_i = 1.$$

The geometry of $F(W)$ is that it gets smaller as the B_i regions get smaller and more uniform in importance (as determined by m_i). Note that $\mathbf{v}_i$ can be thought of as the "average" vector in B_i. If B_i is a small, localized, almost planer region then $\mathbf{v}_i$ will be almost a unit vector that points to the probability-weighted center of B_i. This will make the dot product $\mathbf{w}_i \cdot \mathbf{v}_i$ almost one, since $\mathbf{w}_i$ is a unit vector that also points at the center of B_i. However, if B_i is very large and significantly curved then $\mathbf{v}_i$ will be significantly shorter than a unit vector and the dot product $\mathbf{w}_i \cdot \mathbf{v}_i$ will be smaller, thus increasing $F(W)$. Finally, note that the sum of the $\mathbf{v}_i$ vectors is a constant vector that only depends upon ρ and not on the weight vectors.

$$\sum_{i=1}^{N} \mathbf{v}_i = \int_{\Omega^n} \mathbf{z}\, \rho(\mathbf{z})\, dA(\mathbf{z})$$

In Conclusion, a relatively simple formula exists for the mean squared error $F(W)$ of the CPN network. By virtue of the generality of the approach taken in this definition this formula applies to both the full CPN network and the forward-only version defined below.

4 CPN Variants and Evolutes

Multiple types of counterpropagation network that have been defined; with the CPN module defined above being the most important example. One useful CPN variant is illustrated in Figure 4. In this network layer 4 of the basic CPN network has been excised, as have the interconnects from the **y** input vector to layer 3 (this last change is optional). The net result is a CPN variant (called the *forward-only CPN module*) that is ideal for problems in which only transformations from **x** to **y** are of interest (and implementation of the inverse mapping and completions of partial **x**-**y** input pairs are not of interest).

Figure 5 illustrates a scheme for using CPN modules as building blocks for building a larger network. The idea is that each module receives its **x** input from the **y**′ output of the previous module (except for the first module, which actually receives **x**. Each layer receives the same **y** input, namely the finally desired **y** itself. This scheme is much like a multistage rocket - each CPN lookup table module goes as far as it can towards the final goal (**y**) and then subsequent modules take the process one step further. This is reminiscent of Ivakhnenko's Group Method of Data Handling (GMDH) and Barron's Adaptive Learning Network (ALN) [21].

Finally, by modifying the competition process on layer 3 of the CPN, there can be K winning processing elements (corresponding to the K nearest matching $\mathbf{w}_i$ vectors) instead of one (see [37] for additional discussion about the advantages of this). If the outputs of these K processing elements are set so that they sum to one (i.e., the former single output signal of one is now divided among the K units in some manner dependant upon their relative I_i values) the the outstars of layers 2 and 4 will *blend* their usual output values. This "interpolation" process can lead to significantly increased mapping approximation accuracy for no increase in network size. A CPN using this approach is operating in an *interpolative* mode, as opposed to the *accretive* mode discussed above.

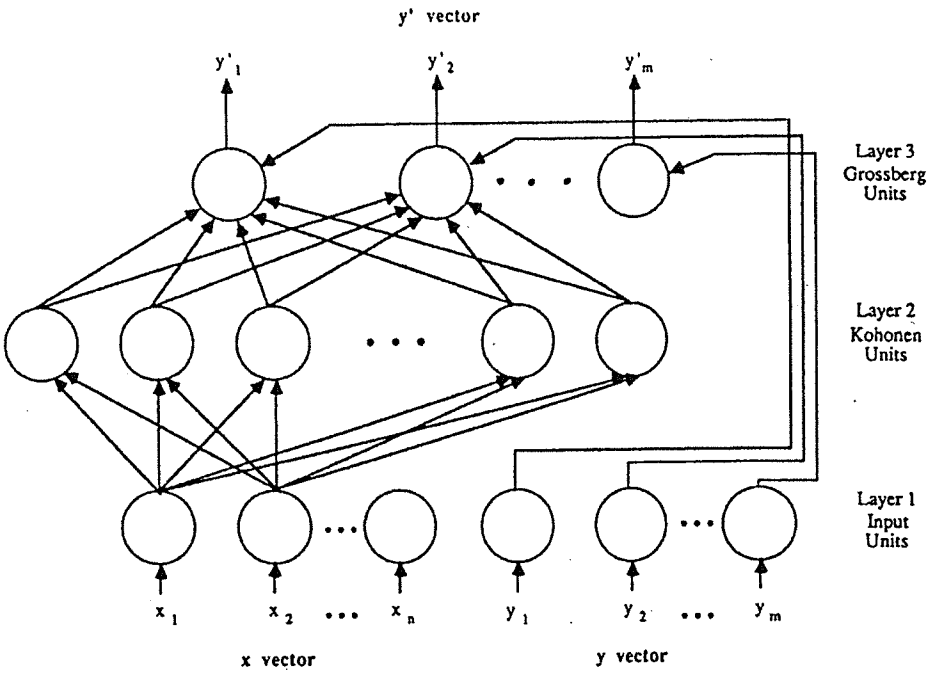

Figure 4: The Forward-Only CPN

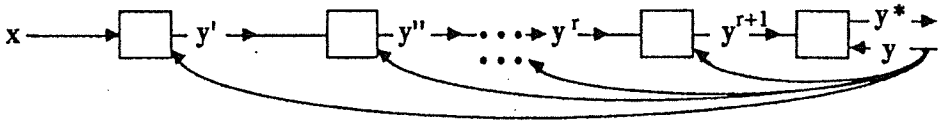

Figure 5: A Multilayer CPN Scheme

5 Discussion

One of the problems with CPN (returning now to the notation of Figures 1 and 2) is that Kohonen learning only works well when ρ is non-zero only in a single connected region. If this is not true then weight vectors can get "stuck" in isolated regions and cannot move to where they are needed. A number of solutions have been developed for this. One is called *convex combination*. In this approach, one starts off with all the (unit) weight vectors equal to the vector $1 = (1/\sqrt{N}, 1/\sqrt{N}, \ldots, 1/\sqrt{N})$. The x-y data inputs then start out as convex combinations (i.e., $\alpha x + (1-\alpha)1$) of their actual values and this same vector. Initially, the combinations start off with a very low value of α (near zero). This forces all of the data vectors to be close to the weight vectors. As time goes on the value of α is raised slowly to one. As this happens, the weight vectors are "peeled off" and follow the input data vectors as they move away from 1. This works very well but it slows down adaptation. Another approach is to add noise to the data, which has the effect of making ρ positive everywhere. This works all right but is much slower than convex combination. Finally, another approach is to build a "conscience" into each Kohonen processing element to monitor its success in the layer 3 competition. If it wins the competition substantially more often than $1/N$ of the time, then have that unit take itself out of the competition for a while, allowing stuck units to win their way out of captivity. Initial work suggests that this approach works well and may solve the problems with Kohonen's Proposition 5.1. This idea of adding a conscience to the layer 3 processing elements is due to Duane Desieno of Logical Designs in San Diego.

Given its simple lookup table function, CPN is obviously inferior to backpropagation for most mapping network applications. Its advantages are that it is simple and that it forms a good statistical model of its input vector environment (i.e., of ρ). Another potential advantage is the existence of a closed-form formula for the mean squared error of the network's mapping function.

As with all lookup table approaches, CPN requires a large number of entries (Kohonen units) in order to achieve high mapping approximation accuracy. However, because CPN adjusts its weights to conform to the statistics of the input vectors the number of Kohonen processing elements required to reach a particular level of mapping approximation accuracy depends primarily on the complexity of the mapping and the statistics of the x and y vector selections, and not on the dimensionality of the spaces involved. Thus, CPN avoids the worst part of Bellman's "curse of high dimensionality". The use of CPN in an interpolation mode can also help accuracy considerably.

CPN seems to be well suited to situations where the relationships between the input data components is more important than the implementation of a mapping (although it does fairly well as a mapping network). The network is also useful for rapid prototyping of neurocomputing systems (even if a backpropagation or some other network will be used in its place later after the rest of the system is working) because it typically converges one or two orders of magnitude faster than backpropagation. The advantages of multilayer CPN module schemes such as that of Figure 5 have not yet been assessed.

Finally, CPN illustrates a key point about neurocomputing system design. Namely, that many of the existing network paradigms can be viewed as building block components that can be assembled into new configurations that offer different information processing capabilities.

References

[1] Amari, Shun-ichi, "Field Theory of Self-Organizing Neural Networks", *IEEE Trans. on Sys., Man and Cyber.*, SMC-13, No. 5, 741-748, Sept/Oct 1983.

[2] Amari, Shun-ichi, and Arbib, Michael A., **Competition and Cooperation in Neural Nets**, Springer-Verlag, 1982.

[3] Amari, Shun-ichi, "A Method of Statistical Neurodynamics", *Biological Cybernetics*, 14, 201-225, 1974.

[4] Amari, Shun-ichi, "Characteristics of Randomly Connected Threshold-element Networks and Network Systems", *Proc. IEEE*, 59, No. 1, 35-47, January 1971.

[5] Anderson, James A., Golden, Richard M., and Murphy, G.L., "Concepts in Distributed Systems", *SPIE Proc.*, 634, 260-276,1986.

[6] Anderson, James A., "Cognitive and Psychological Computation with Neural Models", *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13, No. 5, September/October 1983.

[7] Anderson, James A., "A Simple Neural Network Generating an Interactive Memory", *Math. Biosci.*, 14, 197-220, 1972.

[8] Barto, Andrew G., and Sutton, Richard S., **Simulation Experiments With Goal-seeking Adaptive Elements**, AFWAL-TR-84-1022 (DTIC Doc. No. ADA 140295), February 1984.

[9] Barto, Andrew G., and Sutton, Richard S., "Landmark Learning: an Illustration of Associative Search", *Biological Cybernetics*, 42, 1-8, 1981.

[10] Carpenter, Gail A., and Grossberg, Stephen, "A Massively Parallel Architecture for a Self-organizing Neural Pattern Recognition machine", *Computer Vision, Graphics and Image Processing*, 37, 54-115, 1987.

[11] Casasent, David, **CMU Optical Processing Research for Scene Analysis**, *SPIE Proc.*, 634, 1986.

[12] Cohen, Michael A., and Grossberg, Stephen, "Neural Dynamics of Speech and Language Coding: Developmental Programs, Perceptual Grouping, and Competition for Short Term Memory", *Human Neurobiology*, 5, 1-22, April 1986.

[13] Cohen, Michael A., and Grossberg, Stephen, "Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks", *IEEE Transactions On Systems, Man and Cybernetics*, SMC-13, No. 5, 815-826, September/October 1983.

[14] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification", *IEEE Trans. Infor. Theory*, IT-13, 21-27, January 1967.

[15] Cruz, Claude, and Tam, J.Y., **NEP: An Emulation-Assist Processor for Parallel Associative Networks**, IBM Palo Alto Scientific Center Report Number G320-3475, 1985.

[16] Cruz, Claude, and Myers, H.J., **Associative Networks II**, IBM Palo Alto Scientific Center Report Number G320-3446, 1983.

[17] Cruz, Claude, and Myers, H.J., **Associative Networks**, IBM Palo Alto Scientific Center Report Number G320-3474, 1982.

[18] Daugman, J. G., "Uncertainty Relation for Resolution in Space, Spatial Frequency, and Orientation Optimized by Two-dimensional Visual Cortical Filters", *J. Opt. Soc. Am.*, Vol. A, No. 2, pp. 1160-1169, July 1985.

[19] Denker, John, **Proc. Second Annual Conference on Neural Networks for Computing**, American Institute of Physics, Proceedings Vol. 151, 1986.

[20] Dunning, G.J., Marom, E., Owechko, Yuri, and Soffer, Bernard H., "Optical Holographic Associative Memory Using a Phase Conjugate resonator", *Proc. SPIE*, **625**, 1986.

[21] Farlow, Stanley J. (Ed.), **Self-Organizing Methods in Modeling: GMDH Type Algorithms**, Marcel Dekker, 1984.

[22] Fisher, Arthur D., Fukuda, Robert C., and Lee, John N., "Implementations of Adaptive Associative Optical Computing Elements", *Proc. SPIE*, **625**, 1-5, 1986.

[23] Fisher, Arthur D., and Giles, C. Lee, "Optical Adaptive Associative Computer Architectures", *Proc. IEEE COMPCOM Meeting* (IEEE Cat. No. CH2135-2/85), pp. 342-344, February 1985.

[24] Fisher, Arthur D., Giles, C. Lee, and Lee, John N., "Associative Processor Architectures for Optical Computing", *J. Optical Soc. Am. ,A*, **1**, 1337-, 1984.

[25] Fukushima, K., and Miyake, S., "Neocognition: A New Algorithm for Pattern Recognition Tolerant of Deformations and Shifts in Position", *Pattern Recognition*, **15**, No. 6, 455-469, 1984.

[26] Geman, Stuart "The Law of Large Numbers in Neural Modeling", in Grossberg, S. [Ed.], **Mathematical Psychology and Psychophysiology**, American Mathematical Society, 1981.

[27] Grossberg, Stephen, and Stone, Gregory, "Neural Dynamics of Word Recognition and Recall; Attentional Priming, Learning, and Resonance", *Psychological Review*, 93, No. 1, 46-74, 1986.

[28] Grossberg, Stephen, and Kuperstein, Michael, **Neural Dynamics of Adaptive Sensory-Motor Control**, North-Holland, 1986.

[29] Grossberg, Stephen, and Mingolla, Ennio, "Neural Dynamics of Perceptual Grouping: Textures, Boundaries, and Emergent Segmentations", *Perception & Psychophysics*, 38, No. 2, 141-171, 1985.

[30] Grossberg, Stephen, and Michael A. Cohen, **Some Global Properties of Binocular Resonances: Disparity Matching, Filling-In, and Figure-Ground Synthesis**, in Caelli, T., and P. Dodwell, P. [Eds.], **Figural Synthesis**, , Earlbaum, 1984.

[31] Grossberg, S., **Studies of Mind and Brain**, Reidel, 1982.

[32] Grossberg, S., "Embedding Fields: Underlying Philosophy, Mathematics, and Applications to Psychology, Physiology, and Anatomy", *J. Cyber*, 1, 28-50, 1971.

[33] Grossberg, S., "Some Networks That Can Learn, Remember, and Reproduce Any Number of Complicated Space-time Patterns, II", *Stud. App. Math.*, 49, 135-166, 1970.

[34] Grossberg, S., "Embedding Fields: a Theory of Learning With Physiological Implications", *J. Math Psych.*, 6, 209-239, 1969.

[35] Grossberg, S., "Some Networks That Can Learn, Remember, and Reproduce Any Number of Complicated Space-time Patterns, I", *J. Math. & Mech*, 19, 53-91, 1969.

[36] Grossberg, S., "Nonlinear Difference-differential Equations in Prediction and Learning Theory", *Proc. Nat. Acad. Sci.*, 58, 1329-1334, 1967.

[37] Hecht-Nielsen, Robert, "Combinatorial Hypercompression", *Proc. IEEE International Conference on Neural Networks - 1987.*

[38] Hecht-Nielsen, Robert, "Kolmogorov's Mapping Neural Network Existence Theorem", *Proc. IEEE International Conference on Neural Networks - 1987.*

[39] Hecht-Nielsen, Robert, "Neurocomputer Applications", *Proc. National Computer Conference - 1987*, American Federation of Information Processing Societies, 1987.

[40] Hecht-Nielsen, Robert, "Neural Network Nearest Matched Filter Classification of Spatiotemporal Patterns", *Applied Optics*, 15 May 1987.

[41] Hecht-Nielsen, Robert, **Performance Limits of Optical, Electro-Optical, and Electronic Artificial Neural System Processors**, *Proc. SPIE*, 634, 277-306, 1986.

[42] Hecht-Nielsen, Robert, Book Review of Grossberg's "Studies of Mind and Brain", *J. Math. Psych.*, 27, No. 3, 335-340, 1983.

[43] Hecht-Nielsen, Robert, "Neural Analog Processing", *Proc. SPIE*, 360, 180-189, 1982.

[44] Hecht-Nielsen, Robert, "Neural Analog Information Processing", *Proc. SPIE*, 298, 138-141, 1981.

[45] Hinton, Geoffrey E., and Anderson, James A. (Eds.), **Parallel Models of Associative Memory**, Erlbaum, 1981.

[46] Hopfield, J. J., and Tank, D. W., "Neural Computation of Decisions in Optimization Problems", *Biological Cybernetics*, 52, 141-152, July 1985.

[47] Hopfield, J. J., "Neurons With Graded Response Have Collective Computational Properties Like Those of Two-state Neurons", *Proc. Natl. Acad. Sci.*, 81, 3088-3092, May 1984.

[48] Hopfield, J. J., "Neural Networks and Physical Systems With Emergent Collective Computational Abilities", *Proc. Nat. Acad. Sci. USA*, 79, 2554-2558, April, 1982.

[49] Klopf, A. Harry, **The Hedonistic Neuron**, Hemisphere Press, Washington, D.C., 1982 .

[68] Sejnowski, Terrence J., **Skeleton Filters in the Brain**, in Hinton, Geoffrey E., and Anderson, James A. [Eds.], **Parallel Models of Associative Memory**, Erlbaum, 1981.

[69] Soffer, Bernard H., Dunning, G.J., Owechko, Y., and Marom, E., "Associative Holographic Memory with Feedback Using Phase- Conjugate Mirrors", *Optics Letters*, **11**, 118-120, February 1986.

[70] Steinbuch, K., and Piske, U. A. W., "Learning Matrices and Their Applications", *IEEE Trans. on Elec. Computers*, **12**, December 1963.

[71] Steinbuch, K., "Die Lernmatrix", *Kybernetik (Biol. Cyber.)*, **1**, 36-45, 1961.

[72] Thakoor, Anil, **Content-Addressable, High Density Memories Based on Neural Network Models**, JPL Report D-4166, March 1987.

[73] Widrow, Bernard, and Stearns, Samuel D., **Adaptive Signal Processing**, Prentice-Hall, 1985.

[74] Widrow, Bernard, "Generalization and Information Storage in Networks of ADALINE Neurons" in Yovitts, G. T., **Self-Organizing Systems**, Spartan Books, 1962.

[75] Widrow, Bernard, and Hoff, M., Jr., "Adaptive Switching Circuits", *IRE WESCON Conv. Record, Part 4*,96-104, 1960.

[76] Willshaw, D. J., **Models of Distributed Associative Memory**, Ph.D. Thesis, University of Edinburgh, 1971.

[77] Willshaw, D.J. and Longuet-Higgins, H.C., "Associative Memory Models", in **Machine Intelligence**, B. Meltzer and O. Michie, Edinburgh: Edinburgh University Press, 1970.

[78] Willshaw, D.J., Buneman, O.P. and Longuet-Higgins, H.C., "Non-holographic Associative Memory", *Nature*, 222,960-962, June 1969.

[79] P.M. Woodward, **Probability and Information Theory with Applications to Radar**, Pergamon Press, 1953.