Alternating Projection Neural Networks

ROBERT J. MARKS II, SENIOR MEMBER, IEEE, SEHO OH, AND LES E. ATLAS, MEMBER, IEEE

Abstract —**We consider a class of neural networks whose performance can be analyzed and geometrically visualized in a signal space environment. Alternating projection neural networks (APNN's) perform by alternatively projecting between two or more constraint sets. Criteria for desired and unique convergence are easily established. The network can be configured as either a content addressable memory or classifier. Convergence of the APNN can be improved by the use of sigmoid-type nonlinearities and/or increasing the number of neurons in a hidden layer.**

I. Introduction

IN THIS PAPER, we depart from the performance analysis techniques normally applied to neural networks. Instead, a signal space approach is used to gain new insights via ease of analysis and geometrical interpretation. Building on a foundation laid elsewhere [1]–[3], we demonstrate that alternating projection neural network's (APNN's) formulated from such a viewpoint can be configured in layered form as a classifier or homogeneously as a content addressable memory [4].

The neurons in the homogeneous APNN can be clamped to a preassigned value and provide the network stimulus or can float in accordance to the stimulus of other neurons. The status of a neuron as clamped or floating may change from application to application. The APNN in this form acts as a content addressable memory. After being trained with a number of library vectors, the APNN can reconstruct any one library vector by clamping an arbitrary subset of the neurons to values equal to the elements of that vector. If the cardinality of the subset is sufficiently large, the states of the remaining floating neurons will converge to the unknown vector elements.

The neurons in the input layer of the layered APNN provide the network's stimulus. Use of neurons in the hidden layer increase storage capacity, convergence rate and classification diversity. When used as a classifier, the states of the output neurons provide the classification index.

APNN's have the following attributes:

(a) As their name suggests, APNN's perform by alternatingly projecting between two or more constraint sets. This is in contrast, for example, to the more conventional technique of formulation of an energy metric for the neural networks, establishing a lower energy bound and showing that the energy reduces each iteration [5]–[11]. The accuracy of the steady-state solution of APNN's can be easily assured for both synchronous and asynchronous operation.

(b) Many homogeneous neural networks [9], [12]–[15] do not scale well, i.e., the storage capacity less than doubles when the number of neurons is doubled [16], [17]. We show that, in layered form, the number of stored patterns in an APNN is roughly equal to the number of input and hidden neurons.

(c) The layered APNN differs from a backpropagation neural network [18], [19] in that each item of training data is stored in an exact form, i.e., in the absence of computational inexactness, the classification performance of the network on the training data will be perfect. This type of behavior is useful in applications, such as "memory-based reasoning" [20], where large and accurate databases are needed. These types of applications are not well suited to conventional parallel architectures, since increasing the number of processors also increase the computational time.

The outline of this paper is as follows. After a brief review of convex set projection theory and establishment of the dynamics of the APNN, we present proofs of convergence for both synchronous and dispersionless asynchronous operation. Sufficient criteria for proper convergence are established. The convergence dynamics of the APNN are explored and illustrated geometrically. Effects of noncompliance with required convergence criteria and learning are also geometrically interpreted.

II. An Overview of POCS

The technique of projection onto convex sets (POCS) [21], [22] has traditionally been applied to signal restoration. In this section, we give a brief overview of POCS. A more in depth treatment is in the book by Stark [22].

A set of vectors, $\mathscr{C}$, is said to be convex if, for all vectors $\vec{x}$ and $\vec{y}$ in $\mathscr{C}$,

$$(1-\alpha)\vec{x}+\alpha\vec{y}\in\mathscr{C}, \qquad 0\leqslant\alpha\leqslant 1.$$

Geometrically, this is interpreted as requiring the line segment connecting $\vec{x}$ and $\vec{y}$ be totally subsumed in $\mathscr{C}$. Examples include subspaces (planes), boxes, balls and linear varieties (translated subspaces).

As illustrated in Fig. 1, the projection of an arbitrary vector, $\vec{g}$, onto a convex set results in the unique vector in $\mathscr{C}$ closest to $\vec{g}$ in the mean square sense. Consider, then, I

Manuscript received November 10, 1987; revised July 25, 1988 and February 15, 1989. This work was supported by the Washington Technology Center and by SDIO/IST through the U.S. Office of Naval Research in conjunction with the Optical Systems Laboratory at Texas Tech University, Lubbock, TX. L. E. Atlas was supported in part by a National Science Foundation Presidential Young Investigator Award.

The authors are with the Interactive Systems Design Laboratory, Department of Electrical Engineering, University of Washington, Seattle, WA 98195.

IEEE Log Number 8927432.

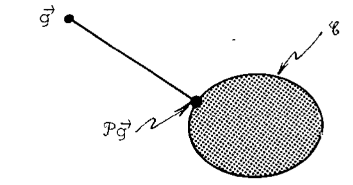


Fig. 1. The two-dimensional shaded region is a convex set, $\mathscr{C}$. For an arbitrary vector $\vec{g}$, the projection onto the convex set, $\mathscr{P}\vec{g}$, is that point in $\mathscr{C}$ closest to $\vec{g}$.

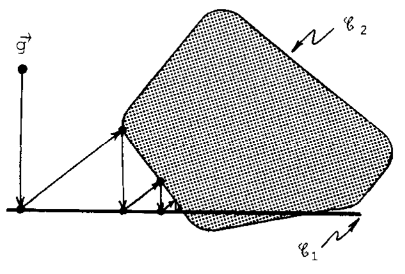


Fig. 2. Illustration of POCS (projection onto convex sets). Alternating projections between the (convex) line segment, $\mathscr{C}_1$, and the shaded convex set, $\mathscr{C}_2$, asymptotically approaches a point common to their intersection.

convex sets with common (convex) intersection $\mathscr{C}$:

$$\mathscr{C}_1 \cap \mathscr{C}_2 \cap \cdots \cap \mathscr{C}_I = \mathscr{C} \neq \varnothing .$$

As is illustrated in Fig. 2, the fundamental result of POCS is that repeated sequential projection onto these sets asymptotically approaches a point in $\mathscr{C}$. The design of the APNN is based on POCS. The network iterates between convex sets the single point intersection of which is a desired steady-state solution.

III. The Alternating Projection Neural Network

In this section, we established the notation for the APNN. Nonlinear modifications to the network made to impose certain performance attributes are considered later.

Consider a set of N continuous level linearly independent library vectors (or patterns) of length $L > N$: $\{\vec{f}_n | 0 \leqslant n \leqslant N\}$. We form the library matrix

$$F = \left[\vec{f}_1 \mid \vec{f}_2 \mid \cdots \mid \vec{f}_N\right]$$

and for reasons soon to be made clear, choose the neural network interconnect matrix [1]–[3], [23], [24]

$$T = F(F^T F)^{-1} F^T \tag{1}$$

where the superscript T denotes transposition. The interconnect value between neurons p and k is t_{pk}. Since T is symmetric, $t_{pk} = t_{kp}$. We divide the L neurons into two sets: one in which the states are known and the remainder in which the states are unknown. This partion may change from application to application. Let $s_k(M)$ be the state of the kth neuron at time M. If the kth neuron falls into the known category, its state is *clamped* to the known value (i.e., $s_k(M) = l_k$ where $\vec{l}$ is some library vector). We first consider the case where the remaining floating neurons are equal to the sum of the inputs into the node. That is, $s_k(M) = i_k$, where

$$i_k = \sum_{p=1}^{L} t_{pk} s_p. \tag{2}$$

If all neurons change state simultaneously (i.e., $s_p = s_p(M-1)$), then the net is said to operate synchronously. If only one neuron changes state at a time, the network is operating asynchronously.

Let P be the number of clamped neurons. We will prove that the neural states converge strongly to the library vector corresponding to the P clamped neurons if the first P rows of F (denoted F_P) form a matrix of full column rank. That is, no column of F_P can be expressed as a linear combination of those remaining. By strong convergence,[1] we mean

$$\lim_{M \to \infty} \|\vec{s}(M) - \vec{f}\| = 0$$

where $\|\vec{x}\|^2 = \vec{x}^T \vec{x}$. Proof of this proposition is in Section IV. Techniques to improve the network's convergence rate are in Section IV.

Lastly, note that subsumed in the criterion that F_P be full rank is the condition that the number of library vectors not exceed the number of clamped neural states $(P \geqslant N)$. Techniques to bypass this restriction by using hidden neurons are discussed in Section VII.

Example

A total of $N = 4$ library vectors of length $L = 25$ were produced by a uniform random number generator. A plot of the vector $\vec{f}_1$ is shown in Fig. 3(a). The interconnect matrix in (1) was formed. The last $P = 15$ elements of $\vec{f}_1$ were used as the clamped values of the APNN. Shown in Fig. 3(b)–(d) are the states of the floating neurons for $M = 2$, 5 and 20 synchronous iterations. Clearly, $\vec{s}(20) \approx \vec{f}_1$ and the remainder of the library vector has been resurrected.

Partition Notation

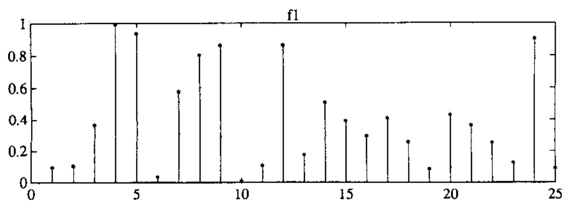
In the homogeneous form of the APNN, the partition of clamped and floating neurons can change from application to application. For a given application, however, we can assume without loss of generality that neurons 1 through P are clamped and the remaining neurons are floating. We adopt the vector partitioning notation

$$\vec{i} = \left[\begin{array}{c} \vec{i}^P \\ \hline \vec{i}^Q \end{array}\right]$$

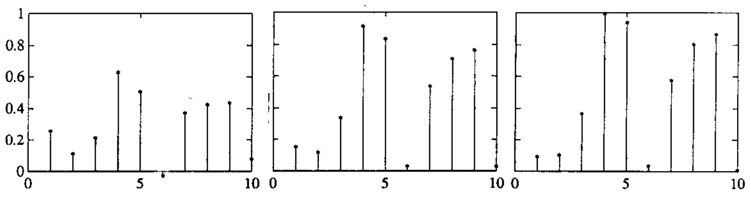
where $\vec{i}^P$ is the P-tuple of the first P elements of $\vec{i}$ and $\vec{i}^Q$ is a vector of the remaining $Q = L - P$. We can thus write, for example,

$$F_P = \left[\vec{f}_1^P | \vec{f}_2^P | \cdots | \vec{f}_N^P\right].$$

[1]The convergence proofs to be referenced later prove strong convergence in an infinite dimensional Hilbert space. In a discrete finite dimensional space, however, both strong and weak [21], [22] convergence imply that $\vec{s}(M) \to \vec{f}$ as $M \to \infty$.



(a)



(b) (c) (d)

Fig. 3. Illustration of homogeneous APNN results. Four library vectors of length $L = 25$ were stochastically generated. (a) Values of the vector $\vec{f}_1$. The states of the final 15 values of $\vec{f}_1$ were used as the clamped neural values in an APNN. (b)–(d) The states of the first ten floating neurons, after $M = 2$, 5, and 20 synchronous iterations, respectively. The first ten elements of $\vec{f}_1$ have been clearly reconstructed in (d).

Using this partition notation, we can define the neural clamping operator by

$$\eta\vec{i} = \left[\frac{\vec{f}^P}{\vec{i}^Q}\right]$$

Thus the first P elements of $\vec{i}$ are clamped to $\vec{f}^P$. The remaining Q nodes "float."

Partition notation for the interconnect matrix will also prove useful. Define

$$T = \left[\begin{array}{c|c} T_2 & T_1 \\ \hline T_3 & T_4 \end{array}\right]$$

where T_2 is a P by P and T_4 a Q by Q matrix. The subscripts are motivated by quadrant location. Some properties of T and T_4 are discussed in Appendix A.

IV. Steady-State Convergence Proofs

In this section, we provide convergence of the APNN's for synchronous operation. If stability is assumed, convergence occurs when the time delay between each neuron pair is fixed yet varies from pair to pair. Each proof requires that F_P be full rank. The behavior of the network when F_P is not full rank also addressed.

Synchronous Operation

For synchronous operation, the network iteration in (2) can be written as

$$\vec{i}\,(M) = T\vec{s}\,(M).$$

The known (or clamped) neural states are then imposed to generate the updated state vector

$$\vec{s}\,(M+1) = \eta\vec{i}\,(M).$$

Thus the iterative state equation can be written as

$$\vec{s}\,(M+1) = \eta T\vec{s}\,(M) \tag{3}$$

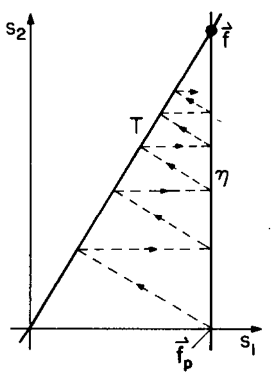


Fig. 4. The linear variety, η, and the subspace T intersect at the library vector, $\vec{f}$. By alternatly projecting between the subspace and linear variety, the neural network is seen to converge to a point common to both.

As is illustrated in Fig. 4, this operation can easily be visualized in an L-dimensional signal space. The T matrix orthogonally projects any vector onto a N-dimensional subspace, T, formed by the closure of the library vectors [25]. (Kohonen [26] has suggested a single projection onto T as an associative memory algorithm.) The clamping operator, η, orthogonally projects onto the Q-dimensional linear variety, η, formed by the set of all L tuplets with their first P elements equal to $\vec{f}^P$. According to POCS, alternating orthogonal projection between these two convex sets strongly converges to a point common to both. (When the $I = 2$ convex sets are linear varieties as they are here, the algorithm is equivalent to Von Neumann's alternating projection theorem [27], [28].) Clearly, the library vector $\vec{f}$ is common to both T and η. The requirement that F_P has full column rank assures that $\vec{f}$ is the *only* point of intersection [1] and our proof is complete. Note that the network will properly converge for any initialization of the floating neuron states.

Convergence Solution

For a given partition with P clamped neurons, (3) can be written in partitioned form as

$$\left[\frac{\vec{f}^P}{\vec{s}^Q(M+1)}\right]=\eta\left[\begin{array}{c|c} T_2 & T_1 \\ \hline T_3 & T_4 \end{array}\right]\left[\frac{\vec{f}^P}{\vec{s}^Q(M)}\right]. \tag{4}$$

The states of the P clamped neurons are not affected by their input sum. Thus there is no contribution to the iteration by T_1 and T_2. We can equivalently write (4) as

$$\vec{s}^Q(M+1)=[T_3|T_4]\left[\frac{\vec{f}^P}{\vec{s}^Q(M)}\right]$$

or

$$\vec{s}^Q(M+1)=T_3\vec{f}^P+T_4\vec{s}^Q(M). \tag{5}$$

We show in Appendix A that if F_P is full rank, then the spectral radius (magnitude of the maximum eigenvalue) of T_4 is strictly less than one. It follows that the steady-state solution of the difference equation in (5) is

$$\vec{f}^Q=(I-T_4)^{-1}T_3\vec{f}^P \tag{6}$$

where, since F_P is full rank, we have made use of our previous observation that

$$\vec{s}^Q(\infty)=\vec{f}^Q. \tag{7}$$

That is, the steady-state solution is the extrapolation of the library vector

$$\vec{f}=\left[\frac{\vec{f}^P}{\vec{f}^Q}\right].$$

Equation (6) could be used in lieu of the APNN *if* the status of each neuron as clamped or floating remained the same from application to application. If the partition changes, however, so does T_3 and T_4. Even if the partition remains static, (6) is not amenable to learning sequentially presented library vectors. We show in Section VI that the interconnect matrix in (1) is.

Clock Skew

If we assume that the APNN reaches a stable solution, then nondispersive clock skew does not affect the steady-state result [29], [30]. Let τ_{kp} denote the time delay between floating neurons k and p and let γ_{kp} denote the time delay between the pth clamped neuron and the kth floating neuron. Then, using our partition convention, the skewed iteration in (2) becomes

$$s_k\{t\}=\sum_{p=1}^{P}t_{kp}f_p^P\mu\{t-\gamma_{kp}\}+\sum_{p=P+1}^{L}t_{kp}s_p\{t-\tau_{kp}\},\qquad P<k\leqslant L$$

where $\mu\{\cdot\}$ denotes the unit step and we have used brackets to denote continuous rather than discrete time. Such a relationship arises, for example, when the time delay among neurons is proportional to their physical separation. Letting $t\to\infty$ and assuming a stable solution gives

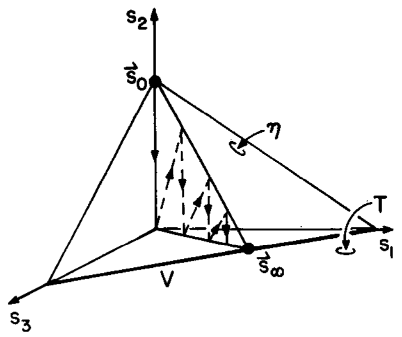


Fig. 5. Illustration of the case where the subspace T and linear variety, η, intersect along the linear variety $\mathcal{V}$. In such underdetermined cases, the APNN will iteratively converge to that point on $\mathcal{V}$ closest to the initialization. For example, as shown, for an initialization of $\vec{s}(0)$, convergence is to $\vec{s}(\infty)$.

$$s_k\{\infty\}=\sum_{p=1}^{P}t_{kp}f_p^P+\sum_{p=P+1}^{L}t_{kp}s_p\{\infty\},\qquad P<k\leqslant L$$

or, in matrix–vector form

$$\vec{s}^Q\{\infty\}=T_3\vec{f}_p^P+T_4\vec{s}^Q\{\infty\}.$$

Since T_4 is not singular if F_P is of full column rank (Theorem 1 in Appendix A), we conclude that the solution to this equation is unique and, from (6), is given by $\vec{s}^Q\{\infty\}=\vec{f}^Q$.

Results of Noncompliance with Conversion Criteria

(a) The Underdetermined Case:

If F_P is not of full column rank, the intersection of the linear varieties η and T results in a linear variety, $\mathcal{V}$, of positive dimension. (Visualize, for example, two planes intersecting in three space). The neural network, in this case, will converge to that point in $\mathcal{V}$ closest to the initial state vector, $\vec{s}(0)$ [31]. Equivalently, $\vec{s}(\infty)$ is the orthogonal projection of $\vec{s}(0)$ onto $\mathcal{V}$. This result is geometrically illustrated in Fig. 5.

(b) Improper Clamping:

Consider the case where the P clamped neurons are not the first P elements of any library vector. The networks will respond in one of two ways:

(a) If the initialization is a linear combination of the columns of F_P, then $\vec{s}^Q(\infty)$ will be the same linear combination of the columns of F_Q.
(b) Otherwise, the linear variety η formed by the initialization does not intersect the subspace T. As illustrated in Fig. 6, the networks will converge to that point on the linear variety closest to the subspace [32], [33].

When T and η do intersect, the sum of the inputs for the clamped neurons approaches the clamped values. This is not the case for non-intersection. (In Fig. 6, for example, using the input sums as the states for the clamped nodes results in $\vec{u}$ rather than $\vec{s}(\infty)$.) A large deviation between the clamped values and the input sum in steady state at

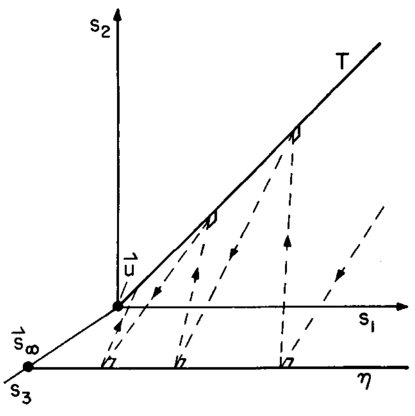


Fig. 6. Two nonintersecting lines are shown in three space. The subspace T is in the (s_1, s_2) plane and the linear variety, η, is on the (s_1, s_3) plane. Alternating orthogonal projections between the two lines iteratively converge to a limit cycle between the two points on each line closest to the other, i.e., $\vec{u}$ and $\vec{s}(\infty)$.

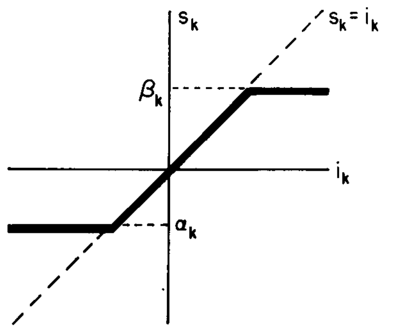


Fig. 7. A saturation nonlinearity. The state of the kth neuron, s_k, is determined by the sum of the inputs to that neuron, i_k.

the clamped neurons thus implies that the network has not been trained with a corresponding library vector.

Neural Saturation as a Convex Constraint

One technique for improving the convergence of the APNN imposes additional convex constraints in the iteration process. Consider, for example, placing a dynamic range constraint on each floating node:

$$s_k = \begin{cases} \alpha_k, & i_k \leqslant \alpha_k \\ i_k, & \alpha_k \leqslant i_k \leqslant \beta_k \\ \beta_k, & i_k \geqslant \beta_k. \end{cases}$$

That is, each node operates linearly between the lower and upper threshold. If the input sum exceeds the upper threshold, β_k, the neural state become β_k. A similar substitution for the lower threshold α_k, is made when appropriate. The resulting nonlinearity shown in Fig. 7 is similar in form to sigmoid nonlinearities used in other neural networks [12], [19].

Neural thresholds can either be predetermined or programmed. If, for example, the library vectors correspond to pixel grey levels, predetermined threshold values can be placed at zero and one. Alternately, the neural thresholds can be programmed during learning. If the kth element of a new vector lies between α_k and β_k, then no change is

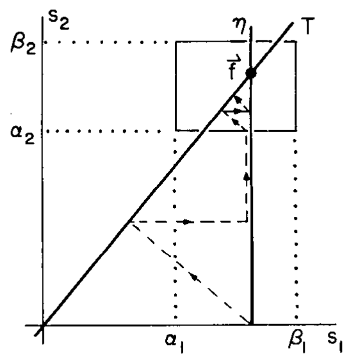


Fig. 8. A geometrical illustration of the effect of the nonlinear neural saturation shown in Fig. 4. In addition to alternately projecting between the subspace, T, and the linear variety, η, projection is also onto a (convex) box the dimensions of which are determined by the neural saturation parameters. Proper convergence will occur if convergence is assured without the box (i.e., F_P is full rank) and the box contains the library element to be restored.

required. If this is not the case, either α_k and β_k are equated to the new value. After training is completed, we have

$$\alpha_k = \min_{1 \leqslant n \leqslant N} f_n(k)$$

and

$$\beta_k = \max_{1 \leqslant n \leqslant N} f_n(k).$$

Upper and lower thresholding of the elements of a vector at preset values can be viewed as the projection of the vector onto a box the sizes of which are specified by the threshold values. As illustrated in Fig. 8, the convergence of the net can be improved by this procedure. Convergence follows immediately from POCS for $I = 3$ convex sets.

Convergence can also be improved by relaxation of the projection operations [1], [4], [21].

V. Training

Direct use of the equation for the interconnect matrix in (1) is generally unacceptable because of the required prior computation of the inverse of a matrix, which, due to the library matrix structure, may be singular or ill conditioned. Furthermore, we desire a technique whereby training data can be incrementally learned in a neural network structure one library vector at a time. Such a procedure for teaching the neural networks new library vectors is developed in this section.

Assume we have an interconnect matrix, $\boldsymbol{T}$, and wish to update the interconnects corresponding to a new library vector, $\vec{f}$. As illustrated in Fig. 9, $\boldsymbol{T}\vec{f}$ projects $\vec{f}$ onto T and

$$\vec{\epsilon} = (\boldsymbol{I} - \boldsymbol{T})\vec{f}$$

is orthogonal to T. The $\vec{\epsilon}$ vector can easily be computed by one synchronous iteration of the net after imposing states equal to $\vec{f}$ on the neurons.

We wish to extend the dimension of the subspace T by one in the direction of $\vec{\epsilon}$. Since $\vec{\epsilon}/\|\vec{\epsilon}\|$ is the unit vector

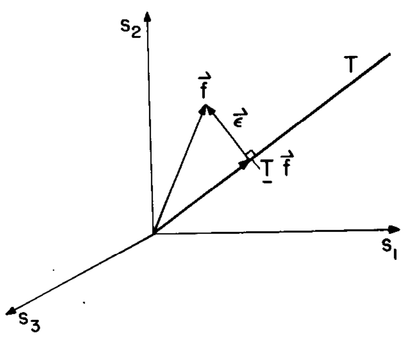


Fig. 9. A geometrical illustration of learning in an APNN. The subspace, T, shown as a line on the (s_1, s_2) plane, is to be augmented to include the new library vector, $\vec{f}$, which also lies in the (s_1, s_2) plane. The error vector, $\vec{\epsilon}$, is determined by the old network's interconnects. The interconnects are updated with this error vector in a Gram–Schmidt procedure. In this illustration, the updated networks interconnects will then project onto the augmented (s_1, s_2) planar subspace.

orthogonal to T, the updated interconnect matrix

$$\boldsymbol{T}^+ = \boldsymbol{T} + \frac{\vec{\epsilon}\,\vec{\epsilon}^{\,T}}{\vec{\epsilon}^{\,T}\vec{\epsilon}}$$

now projects any L-tuple onto the new subspace formed by the closure of T and $\vec{\epsilon}$ or, equivalently, T and $\vec{f}$. The procedure is initiated with all interconnects set to zero. It is similar to that of Gram–Schmidt orthonormalization.

Clearly, if $(\boldsymbol{I} - \boldsymbol{T})\vec{f} = \vec{0}$, the new library vector is already in the subspace T and no updating is required. In practice, computational accuracy will rarely allow an exact equality here. The result is that the dimension of the subspace would be increased in a random manner dictated by computational or other noise. Thus in order to assure the networks is learning something useful, it is thus advisable to compare the error norm $\vec{\epsilon}^{\,T}\vec{\epsilon}$ to some appropriate threshold prior to updating [35]. Note that, as training progresses, the error norm corresponding to any arbitrary vector $\vec{g}$ decreases monotonically. Thus if a training vector is not used to update the interconnect due a small error norm then, after all training is complete, the error norm due to same training vector will not have increased. It thus need not be tested again. (A somewhat different procedure is used in the layered APNN. See Section VII.) Training using such censoring can result in slightly varying T subspaces for different orderings of the library vector. Compounded, this variation can result in a serious degration for large N.

VI. Layered APNN's

The networks thus far considered are homogeneous in the sense that any neuron can be clamped or floating. If the partition is such that the same set of neurons always provides the network stimulus and the remainder respond, then the networks can be simplified. Clamped neurons, for example, ignore the states of the other neurons. The corresponding interconnects can then be deleted from the neural network architecture. When the neurons are so partitioned, we will refer the APNN as *layered.*

In this section, we explore various aspects of the layered APNN and in particular, the use of a so-called hidden layer of neurons to increase the storage capacity of the network. An alternate architecture for a homogeneous APNN that require only Q neurons has been reported [1].

Hidden Layers

In its generic form, the APNN and, indeed, any linear classifier, cannot perform a simple two bit parity check [18], [19], [36]. With the addition of a hidden layer, however, the APNN, can nonlinearly generalize to perform such operations.

Although neural networks will not likely be used for performing parity checks, their use in explaining the role of hidden neurons is quite instructive. The library matrix for two-bit parity is

$$\boldsymbol{F} = \left[\begin{array}{cccc} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ \hline 0 & 1 & 1 & 0 \end{array}\right].$$

The APNN cannot be used to faithfully execute this operation since

$$\boldsymbol{F}_P = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

is not full column rank. Our approach is to augment $\boldsymbol{F}_P$ with two more rows such that the resulting matrix is full rank. Clearly, this can't be accomplished by synthesizing new rows as linear combinations of the first two rows. The column rank would not increase. Most any *nonlinear* combination of the first two rows, however, will in general increase the matrix rank. Such a procedure is potentially applicable to nearly any linear classifier [36]–[41] and, in addition to the layered perceptron, is used in Φ-classifiers [37] and potential function classifiers [39]. Possible nonlinear operations include multiplication, logic operations and running a weighted sum of the clamped neural states through a memoryless nonlinearity such as a sigmoid. This latter alteration is commonly used in neural architectures.

To illustrate with the parity check example, a new hidden neural state is set equal to the exponentiation of the sum of the first two rows. A second hidden neuron will be assigned a value equal to the cosine of the sum of the first two neural states multiplied by $\pi/2$. (The choice of nonlinearities here is arbitrary.) The augmented library matrix is

$$\boldsymbol{F}_+ = \left[\begin{array}{cccc} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ \hline 1 & e & e & e^2 \\ 1 & 0 & 0 & -1 \\ \hline 0 & 1 & 1 & 0 \end{array}\right]. \tag{8}$$

In either the training or look-up mode, the states of the hidden neurons are clamped indirectly as a result of clamping the input neurons.

The playback architecture for this neural network is shown in Fig. 10. The interconnect values for the dashed lines are unity. Clamping the two input neurons then indirectly clamps the hidden neurons using these intercon-

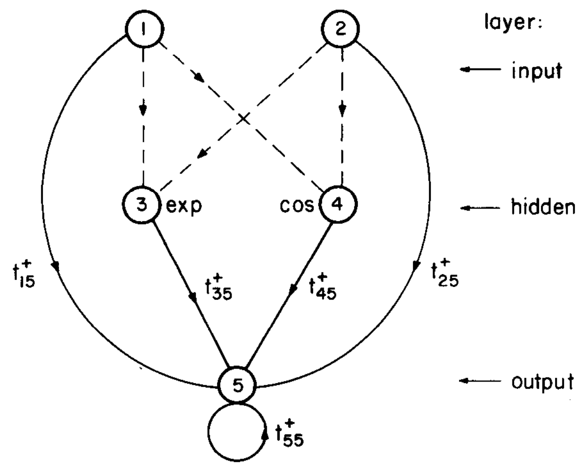


Fig. 10. Illustration of a layered APNN for performing an XNOR. When the states of the input layer are clamped, the hidden layer units are, as a result, also clamped. The transmittance of the dashed line interconnects, in this example, are all unity. The remainder are from the projection matrix values using F_+ in (8) as the library matrix. Numerically, $t_{35} = -t_{15} = -t_{25} = -t_{45} = 0.2471$, and $t_{55} = 0.5753$. In ten synchronous iterations, clamping the inputs to either (0,0) or (1,1) results in an output of 0.00 whereas clamping either to (0,1) or (1,0) results in 0.996.

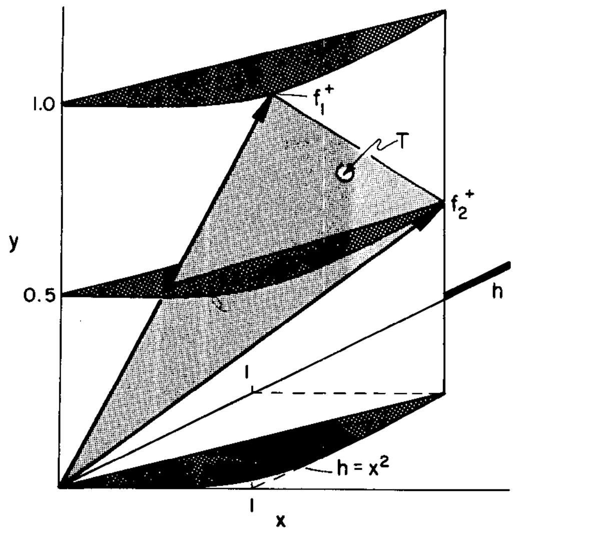


Fig. 11. A geometrical illustration of the use of an x^2 nonlinearity to determine the states of hidden neurons. The library vector $\vec{f}_1^+$ is the only point in the subspace, quadratic surface and the linear variety $x = 1/2$. For $\vec{f}_2^+$, the linear variety corresponding to the clamped neuron is $x = 1$.

nects and the neural nonlinearity. The remaining interconnects are from the projection matrix formed from F_+.

In layered APNN's, sigmoidal saturation nonlinearities can be imposed at each neuron as was done in the homogeneous case.

Geometrical Interpretation

In lower dimensions, the effects of hidden neurons can be nicely illustrated geometrically. Consider the library matrix

$$F = \left[\begin{array}{cc} 1/2 & 1 \\ \hline 1 & 1/2 \end{array}\right].$$

Clearly $F_P = [1/2 \quad 1]$ is not full rank. Let the neurons in the hidden layer be determined by the nonlinearity x^2 where x denotes the elements in the first row of F. Then

$$\begin{aligned} F_+ &= \left[\vec{f}_1^+ | \vec{f}_2^+\right] \\ &= \left[\begin{array}{cc} 1/2 & 1 \\ \hline 1/4 & 1 \\ \hline 1 & 1/2 \end{array}\right]. \end{aligned}$$

The corresponding geometry is shown in Fig. 11 for x the input neuron, y the output and h the hidden neuron. The augmented library vectors are shown and a portion of the generated subspace is shown lightly shaded. The surface of $h = x^2$ resembles a cylindrical lens in three dimensions. Note that the linear variety corresponding to $x = 1/2$ intersects the cylindrical lens and subspace only at $\vec{f}_1^+$. Similarly, the $x = 1$ plane intersects the lens and subspace at $\vec{f}_2^+$. Thus in both cases, clamping the input corresponding to the first element of one of the two library vectors uniquely determines the library vector.

Convergence Improvement

Use of additional neurons in the hidden layer will improve the convergence rate of the APNN. Specifically, the spectral radius of the T_4 matrix is decreased as additional neurons are added. The dominant time constant controlling convergence is thus decreased. A proof is in Appendix B.

Training

If learning in a layered APNN is performed by the previously described Gram–Schmidt procedure, then the network requires intensive interconnection during the learning process since the error, ϵ, at every node is determined by the imposed states of the new library vector at every node. During the recall or playback process, however, the interconnects that provide inputs to the input and hidden layers are not used since these layers are clamped.

When teaching a layered APNN, the error of only the output neurons should be used to determine whether the interconnects need to be updated. Since the error is not being checked at every neuron after one pass through the data, library vectors which were not used to update the interconnects should be rechecked.

Capacity

Under the assumption that nonlinearities are chosen such that the augmented F_P matrix is of full rank, the number of vectors which can be stored in the layered APNN is equal to the sum of the number of neurons in the input and hidden layers. Note, then, that interconnects between the input and output neurons are not needed if there are a sufficiently large number of neurons in the hidden layer to meet a given capacity requirement.

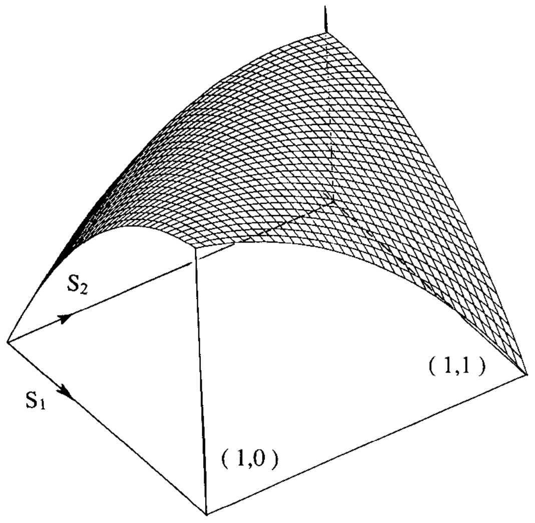


Fig. 12. Response of the elementary parity check APNN using an exponential and trignometric nonlinearity in the hidden layer. Note that, at the corners, the function is equal to the XNOR of the coordinates.

VII. Generalization

We are assured that the APNN will converge to the desired result if a portion of a training vector is used to stimulate the network. What, however, will be the response if an initialization is used that is not in the training set or, in other words, how does the network *generalize* from the training set?

To illustrate generalization, we return to the parity problem. Let $s_5(M)$ denote the state of the output neuron at the Mth (synchronous) iteration. If s_1 and s_2 denote the input clamped value, then

$$s_5(M+1) = t_{15}s_1 + t_{25}s_2 + t_{35}s_3 + t_{45}s_4 + t_{55}s_5(M)$$

where

$$s_3 = \exp(s_1 + s_2) \tag{9}$$

and

$$s_4 = \cos\frac{\pi}{2}(s_1 + s_2). \tag{10}$$

To reach steady state, we let M tend to infinity and solve for $s_5(\infty)$:

$$s_5(\infty) = \frac{1}{1-t_{55}}\left[t_{15}s_1 + t_{25}s_2 + t_{35}\exp(s_1+s_2) + t_{45}\cos\frac{\pi}{2}(s_1+s_2)\right]. \tag{11}$$

A plot of $s_5(\infty)$ versus (s_1, s_2) is shown in Fig. 12. The plot clearly goes through 1 and 0 according to the parity of the corner coordinates. The layered APNN's thus seem to generalize by interpolation. Thresholding Fig. 12 at 3/4 results in the generalization perspective plot shown in Fig. 13.

Note that the equipotential contours in Fig. 12 are all parallel to line $s_1 + s_2 = 0$. This is because the nonlinearities in (9) and (10) are both a function of $s_1 + s_2$ and therefore have the same equipotential contours as $s_1 + s_2 = 0$. Since $t_{15} = t_{25} = -0.2471$, (11) is strictly a function of $s_1 + s_2$. The generalization is therefore dictated by our choice of nonlinearities.

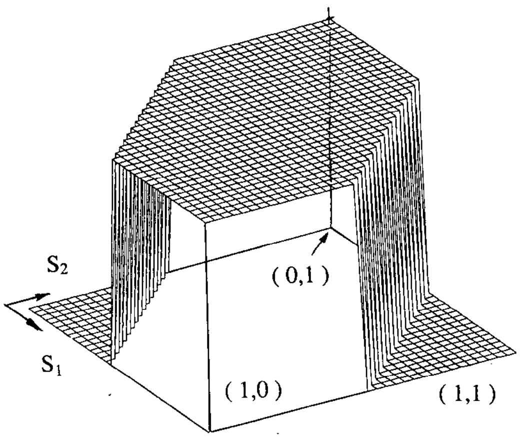


Fig. 13. The generalization of the parity check APNN networks formed by thresholding the function in Fig. 12 at 3/4. Different hidden layer nonlinearities result in different generalizations.

Greater flexibility in classification can be achieved by training the nonlinearities [18], [19] or increasing the number and diversity of hidden layer neurons. We give two examples of the latter approach in which the input and output neurons are not connected. That is, the hidden layer states act as the clamped neurons that provide the stimulus for the floating output neurons. We continue with the parity example, except redefine our library matrix with -1 denoting a logic 0:

$$F = \left[\begin{array}{cccc} -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ \hdashline -1 & 1 & 1 & -1 \end{array}\right].$$

One advantage of this convention is the obvious choice of zero as an output threshold [42].

Spoke Interconnects

For the parity problem, 4 hidden neurons were used. Each used a nonlinearity of $\exp(-z)$. The input-to-hidden interconnect pairs were chosen to be the coordinates from a unit radius circle equally divided into 4 pie slices. The classification generalization shown in Fig. 14 was obtained by thresholding the output at zero and is nearly least mean square. Similar partitions occur using more than 4 spokes. More classification diversity for more complex partitioning requires more spokes.

Stochastic Interconnects

The interconnects between the input and hidden neurons was chosen stochastically [43] from a distribution uniform on $(-1/2, 1/2)$. The nonlinearity was $\exp(-z)$. Generalizations are shown in Fig. 15 for 10 and 50 hidden neurons. The generalization here also seems to approach a least mean square partition.

VIII. Notes

(a) Nonlinearities can also be used to increase the capacity of the homogeneous APNN. Envision, for

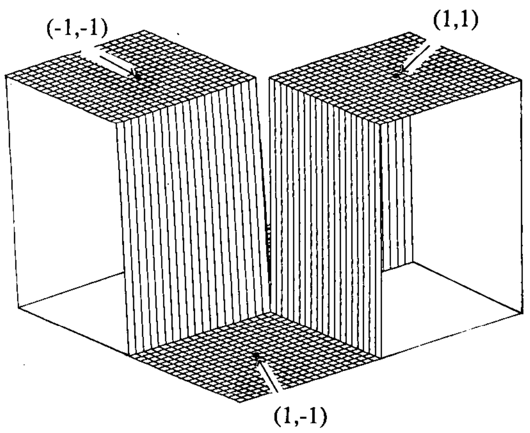


Fig. 14. Generalization of the parity check APNN using 4 hidden neurons and the nonlinearity $\exp(-z)$ for each hidden neuron.

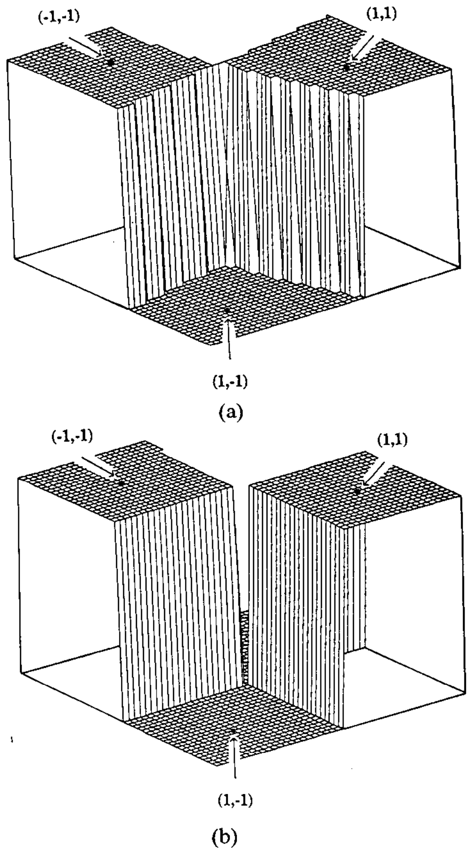


Fig. 15. Generalization of the parity check APNN using 10 (a) and 50 (b) hidden neurons and stochastic interconnects from the input to the hidden layer.

example, associating with each neuron a single hidden neuron whose state is related in some fixed nonlinear manner. Clamping P neurons then clamps an additional P hidden neurons and the network's capacity is essentially doubled.

(b) There clearly exists a great amount of freedom in the choice of the nonlinearities in the hidden layer. Their effect on the network performance is currently not well understood. One can envision, however, choosing nonlinearities to enhance some network attribute such as interconnect reduction, classification region shaping (generalization) or convergence acceleration.

(c) There is a possibility that for a given set of hidden neuron nonlinearities, augmentation of the $\boldsymbol{F}_P$ matrix coincidentally will not result in a matrix of full column rank. Proper convergence is then not assured. Similarly, augmentation may result in a poorly conditioned matrix. In this case, convergence will be quite slow. One obvious practical solution to these problems is to pad the hidden layer with additional neurons. As we have noted, the convergence rate will also improve.

(d) Recently, optical architectures have been proposed for implementing the APNN [44]. Iteration is performed at light speed. The APNN has also been implemented using stochastic processing [45].

Appendix A
Some Properties of T and T_4

Properties of the $\boldsymbol{T}$ *matrix*

The following properties of the projection matrix can be straightforwardly established:

a) $\boldsymbol{T}$ is symmetric and idempotent: from (1), $\boldsymbol{T}=\boldsymbol{T}^2=\boldsymbol{T}^T$.
b) Since $\boldsymbol{T}$ is a projection matrix, its eigenvalues are zero or one.
c) Diagonal elements of $\boldsymbol{T}$ lie between zero and one inclusive.
d) $\operatorname{tr}(\boldsymbol{T})=N=$ number of eigenvalues equal to one.

Definitions ($\boldsymbol{A}$ is a k-by-k matrix):

$\#(\boldsymbol{A})=$ number of eigenvalues of $\boldsymbol{A}$ equal to one

$\nu(\boldsymbol{A})=$ nullity of $\boldsymbol{A}$

$\operatorname{rank}(\boldsymbol{A})=k-\nu(\boldsymbol{A})$.

Lemma 1: Let $\boldsymbol{B}$ denote any real j by k matrix. Then the nonzero eigenvalues of $\boldsymbol{BB}^T$ are the same as those of $\boldsymbol{B}^T\boldsymbol{B}$.

Proof: For any real matrix $\boldsymbol{B}$, there exists orthogonal matrices $\boldsymbol{Q}_L$ and $\boldsymbol{Q}_R$ such that

$$\boldsymbol{B}=\boldsymbol{Q}_L\boldsymbol{V}\boldsymbol{Q}_R.$$

Also, $\boldsymbol{V}$ can be partitioned[2] as

$$\boldsymbol{V}=\left[\begin{array}{c|c}\boldsymbol{D} & \boldsymbol{0}\\ \hline \boldsymbol{0} & \boldsymbol{0}\end{array}\right]$$

where $\boldsymbol{D}$ is a diagonal matrix with no zeros on the diagonal. Thus

$$\boldsymbol{B}^T\boldsymbol{B}=\boldsymbol{Q}_R\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{Q}_R$$

and

$$\boldsymbol{BB}^T=\boldsymbol{Q}_L\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{Q}_L.$$

[2]Dimensionally, the matrix partitions in Appendix A are not the same as those in the *Partition Notation* portion of Section III.

Here

$$V^TV = \left[\begin{array}{c|c} D^2 & 0 \\ \hline 0 & 0 \end{array}\right]$$

is a k by k matrix and

$$VV^T = \left[\begin{array}{c|c} D^2 & 0 \\ \hline 0 & 0 \end{array}\right]$$

is a j by j matrix with the same nonzero eigenvalues.

Lemma 2: All eigenvalues, $\{\lambda_i | 1 \leqslant i \leqslant Q\}$, of T_4 are real and lie in the interval [0,1].

Proof: Let Q denote an orthogonal matrix with the property that

$$QTQ^T = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & 0 \end{array}\right]$$

where I is the N by N identity matrix. We partition Q as

$$Q = \left[\begin{array}{c|c} Q_2 & Q_1 \\ \hline Q_3 & Q_4 \end{array}\right]$$

where Q_2 has dimension N by P. The matrix T_4 can then be written as

$$T_4 = Q_1^TQ_1.$$

Let $S_4 = Q_4^TQ_4$. Then T_4 and S_4 are symmetric matrices and $T_4 + S_4 = I$. Clearly

$$\lambda_i + \lambda_i^s = 1$$

where λ_i and λ_i^s are the eigenvalues of T_4 and S_4, respectively. Because T_4 and S_4 are positive semidefinite matrices,

$$\lambda_i \geqslant 0 \quad \text{and} \quad \lambda_i^s \geqslant 0.$$

The proof follows immediately from these last two equations.

Lemma 3: $\text{rank}(F_2) = \text{rank}(Q_2)$.

Proof: Consider the matrix $G = QF$. We write G as

$$G = \left[\begin{array}{c} G_N \\ \hline G_M \end{array}\right] = QF = Q\left[\begin{array}{c} F_P \\ \hline F_Q \end{array}\right] \tag{A1}$$

where G_N denotes the first N rows of G and we have used the identity

$$\left[G(G^TG)^{-1}G^T\right]G = G.$$

Thus

$$\left[\begin{array}{c|c} I & 0 \\ \hline 0 & 0 \end{array}\right]G = G$$

or, equivalently

$$G_M = 0. \tag{A2}$$

We will show that, as a consequence, G_N is nonsingular. Clearly,

$$\begin{aligned}\det G^TG &= \det F^TQ^TQF \\ &\neq 0\end{aligned}$$

where det denotes the matrix determinant. Thus from (A2)

$$\begin{aligned}\det G^TG &= \det G_N^TG_N \\ &= (\det G_N)^2 \\ &\neq 0\end{aligned}$$

and G_N is not singular.

From (A1), F_P can be written as

$$F_P = Q_2^TG_N.$$

Thus

$$\text{rank}(F_P) = \text{rank}(Q_2^T) = \text{rank}(Q_2)$$

and our proof is complete.

Theorem 1: $\nu(F_P) = \#(T_4) = \nu(I - T_4)$. Thus if F_P is full rank, then the eigenvalues of T_4 are strictly less than one.

Proof:

$$\nu(Q_2Q_2^T) = N - \text{rank}(Q_2Q_2^T) = N - \text{rank}(Q_2).$$

From Lemma 3,

$$\nu(Q_2Q_2^T) = N - \text{rank}(F_P) = \nu(F_P).$$

Since $Q_1Q_1^T + Q_2Q_2^T = I$, and $Q_1Q_1^T$ and $Q_2Q_2^T$ are symmetric, we conclude from Lemmas 1 and 2 that:

$$\#(T_4) = \#(Q_1Q_1^T) = \nu(Q_2Q_2^T) = \nu(F_P)$$

and our proof is completed.

Appendix B
Adding Neurons to the Hidden Layer Improves the Convergence Rate

Partition the augmented library matrix as

$$F_+ = \left[\begin{array}{c} F_P \\ \hline F_H \\ \hline F_Q \end{array}\right] \tag{B1}$$

where F_H contains the state of the hidden layers. Then

$$\begin{aligned}F_+^TF_+ &= F_P^TF_P + F_Q^TF_Q + F_H^TF_H \\ &= B + F_H^TF_H \\ &\equiv A\end{aligned} \tag{B2}$$

where

$$B = F_P^TF_P + F_Q^TF_Q.$$

Manipulation of (B2) gives

$$B^{-1} - A^{-1} = B^{-1}F_H^TF_HA^{-1}. \tag{B3}$$

The augmented interconnect matrix corresponding to (B1) is

$$T_+ = F_+\left(F_+^TF_+\right)^{-1}F_+^T.$$

Let T_4^+ denote the lower right Q by Q partition of T^+. Then

$$T_4^+ = F_QA^{-1}F_Q^T.$$

The spectral radius of T_4^+ dictates the convergence rate of the APNN with hidden neurons. Without hidden neurons, convergence is dictated by the spectral radius of [4]

$$T_4 = F_QB^{-1}F_Q^T.$$

Thus

$$T_4 - T_4^+ = F_Q(B^{-1} - A^{-1})F_Q^T$$
$$= F_Q(B^{-1}F_H^T F_H A^{-1})F_Q^T \qquad \text{(B4)}$$

where, in the second step, we have used (B3).

The right-hand side of (B4) is clearly positive semi-definite. Thus for any vector $\vec{x}$,

$$\vec{x}^T T_4 \vec{x} \geqslant \vec{x}^T T_4^+ \vec{x}.$$

Equality holds if the hidden neurons do not increase the rank of F_P. The spectral radius of T_4^+ is then clearly smaller than that of T_4.

Acknowledgment

The authors express their appreciation to Prof. J. A. Ritcey, D. C. Park, and K. F. Cheung for many insightful discussions.

References

[1] R. J. Marks II, "A class of continuous level associative memory neural nets," *Appl. Opt.*, vol. 26, no. 10, pp. 2005–2010, 1987.

[2] K. F. Cheung, S. Oh, R. J. Marks II, and L. E. Atlas, "Neural net associative memories based on convex set projections," in *Proc. IEEE 1st Int. Conf. on Neural Networks*, vol. III, pp. 245–252, San Diego, CA, 1987.

[3] R. J. Marks II, L. E. Atlas, and K. F. Cheung, "A class of continuous level neural nets," in *Proc. 14th Congress of Int. Commission for Optics Conf.*, pp. 29–30, P.Q., Canada, 1987.

[4] R. J. Marks II, S. Oh, L. E. Atlas, and J. A. Ritcey, "Homogeneous and layered alternating projection neural networks," in *Proc. Int. Symposium on Optical Engineering and Industrial Sensing for Advanced Manufacturing Technologies*, Dearborn, MI, June 1988.

[5] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problem," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

[6] D. W. Tank and J. J. Hopfield, "Simple neural optimization networks: An A/D converter, signal decision circuit and a linear programming circuit," *IEEE Trans. Circuits Syst.*, vol. CAS-33, p. 533, 1986.

[7] M. Takeda and J. W. Goodman, "Neural networks for computation: Number representation and programming complexity," *Appl. Opt.*, vol. 25, no. 18, pp. 3033-3046, 1986.

[8] S. Geman and D. Geman, "Stochastic relaxation, Gibb's distributions, and the Bayesian restoration of images," *IEEE Trans. Patt. Recog. Mach. Intell.*, vol. PAMI-6, pp. 721–741, 1984.

[9] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. National Academy of Sciences*, USA, vol. 79, pp. 2554–2558, 1982.

[10] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Sci.*, vol. 9, pp. 147–169, 1985.

[11] K. F. Cheung, R. J. Marks II, and L. E. Atlas, "Synchronous vs. asynchronous behaviour of Hopfield's CAM neural net," *Appl. Opt.*, vol. 26, no. 22, pp. 4808–4813, 1987.

[12] R. P. Lippman, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, p. 7, Apr. 1987.

[13] N. Farhat, D. Psaltis, and E. Paek, "Optical implementation of the neural model," *Appl. Opt.*, vol. 24, p. 1469, 1985.

[14] L. E. Atlas, "Auditory coding in higher centers of the CNS, "*IEEE Eng. Med. Biol. Mag.*, pp. 29–32, June 1987.

[15] R. J. Marks II and L. E. Atlas, "Geometrical interpretation of Hopfield's content addressable memory," in *Proc. Northcon '88*, Seattle, WA, Oct. 1988.

[16] Y. S. Abu-Mostafa and J. M. St. Jaques, "Information capacity of the Hopfield model," *IEEE Trans. Inform. Theory*, vol. IT-31, p. 461, 1985.

[17] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The capacity of the Hopfield associative memory," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 461–482, 1987.

[18] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing*, vols. I and II, Cambrdige, MA: Bradford Books, 1986.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Willians, "Learning representations by back-propagation errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[20] C. Stanfill and D. Waltz, "Toward memory-based reasoning," *Commun. ACM*, vol. 29, pp. 1213–1228, Dec. 1986.

[21] D. C. Youla and H. Webb, "Image restoration by the method of convex projections: Part I—Theory," *IEEE Trans. Med. Imaging*, vol. MI-1, pp. 81–94, 1982; M. I. Sezan and H. Stark, "Image restoration by the method of convex projections: Part II—Applications and numerical results," *IEEE Trans. Med. Imaging*, vol. MI-1, pp. 95–101, 1985.

[22] H. Stark, *Image Recovery*. New York: Academic 1987.

[23] L. Personnaz *et al.*, "Information storage and retrieval in spin-glass like neural networks," *J. Phys. Lett.*, vol. 46, pp. 359–365, 1985.

[24] J. A. Ritcey *et al.*, "A signal space interpretation of neural nets," in *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 370–372, 1987.

[25] G. Strang, *Linear Algebra and Its Applications*. New York: Academic, 1980, p. 116.

[26] T. Kohonen, *Self-Organization and Associative Memory*. Berlin: Springer-Verlag, 1988, second edition.

[27] J. Von Neumann, *The Geometry of Orthogonal Spaces*. Princeton, NJ Princeton Univ. Press, 1950.

[28] D. W. Montgomery, "Optical applications of Von Neumann's alternating-projection theorem," *Opt. Lett.*, vol. 7, no. 1, pp. 1–3, Jan. 1982.

[29] S. Oh, R. J. Marks II, L. E. Atlas and D. C. Park, "Effects of clock skew in iterative neural networks and optical feedback processors," in *Proc. IEEE Int. Conf. on Neural Networks*, San Diego, CA, July, 1988.

[30] S. Oh, D. C. Park, R. J. Marks II, and L. E. Atlas, "Nondispersive propagation skew in iterative neural network and optical feedback processors," to appear in *Opt. Eng.*

[31] M. R. Civanlar and H. J. Trussel, "Digital signal restoration using fuzzy sets," *IEEE Tran. Acoust., Speech, Signal Processing*, vol. ASSP-34, p. 919, 1986.

[32] M. Goldburg and R. J. Marks II, "Signal synthesis in the presence of an inconsistent set of constraints," *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 647-663, 1985.

[33] D. C. Youla and V. Velasco, "Extensions of a result on the synthesis of signals in the presence of inconsistent constraints," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 465–468, 1986.

[34] V. T. Tom, T. J. Quatieri, M. H. Hayes, and J. H. McClellan, "Convergence of iterative nonexpansive signal reconstruction algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, p. 1052, Oct. 1981.

[35] G. Eichmann and M. Stojancic, "Superresolving signal and image restoration using a linear associative memory," *Appl. Opt.*, vol. 26, no. 10, p. 1911, May 1987.

[36] M. Minsky and S. Papert, *Perceptrons*. Cambridge, MA: MIT Press, 1969.

[37] J. Sklansky and G. N. Wassel, *Pattern Classifiers and Trainable Machines*. New York: Springer-Verlag, 1981.

[38] B. Widrow and M. E. Hoff, "Adaptive Switching Circuits," in *1960 IRE WESCON Convention Record*, pp. 96–104, 1960.

[39] J. T. Tou and R. C. Gonsalez, *Pattern Recognition Principles*, Reading, MA: Addision-Wesley 1974.

[40] F. Rosenblatt, *Principles of Neurodynamics*. Washington, DC: Sparton, 1962.

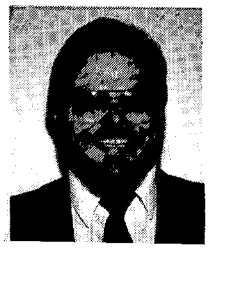
[41] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1972.

[42] R. J. Marks II, J. A. Ritcey, L. E. Atlas, and K. F. Cheung, "Composite Matched Filter Output Partitioning," *Appl. Opt.*, vol. 26, pp. 2274–2278, 1987.

[43] R. J. Marks II, L. E. Atlas, D. C. Park and S. Oh, "The effect of stochastic interconnects in artificial neural network classification," in *Proc. IEEE Int. Conf. on Neural Network*, San Diego, CA, July 1988.

[44] R. J. Marks II, L. E. Atlas, S. Oh, and K. F. Cheung, "Optical Processor Architectures for Alternating Projection Neural Networks," *Opt. Lett.*, vol. 13, pp. 533–535, 1988.

[45] D. Nguyen and F. Holt, "Stochastic processing in a neural network application," in *Proc. IEEE 1st Int. Conf. on Neural Networks*, vol. III, pp. 281–291, San Diego, CA, 1987.



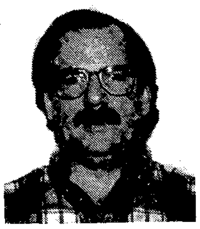
Robert J. Marks II (S'79–M'80–SM'83) joined the faculty of the Department of Electrical Engineering at the University of Washington, Seattle, in December of 1977 where he currently holds the title of Professor. He is Chairman *pro tem* of the IEEE Neural Networks Committee and Chair of the IEEE Circuits & Systems Society Technical Committee on Neural Systems & Applications. He was awarded the Outstanding Branch Councilor Award in 1982 by IEEE and, in 1984, was presented with an IEEE Centennial Medal.

He was a co-founder and first President of the Puget Sound Section of the Optical Society of America and was recently elected that organization's first honorary member. He has over eighty archival journal and proceedings publications in the areas of detection theory, signal recovery, optical computing and artificial neural processing.

Dr. Marks is a member of Eta Kappa Nu and Sigma Xi.

Seho Oh received the B.S. degree in electronics engineering from Seoul National University and M.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology, Seoul. Since September 1986, he has been working toward the Ph.D. degree in electrical engineering at the University of Washington.

From 1981 through 1986, he was with Goldstar Central Research Laboratory in Seoul. His present research interests are signal processing, optical neural nets, and pattern recognition.



Les E. Atlas received the B.S.E.E. degree from the University of Wisconsin and the M.S. and Ph.D. degrees from Stanford University, CA.

He joined the University of Washington Engineering in 1984 and is currently an Associate Professor of Electrical Engineering. He co-founded the Interactive Systems Design Laboratory at University of Washington and he is currently doing research in speech processing and recognition, neural network classifiers, and biologically inspired signal processing algorithms and architectures.

Dr. Atlas was a 1985 recipient of a National Science Foundation's Presidential Young Investigator Award.