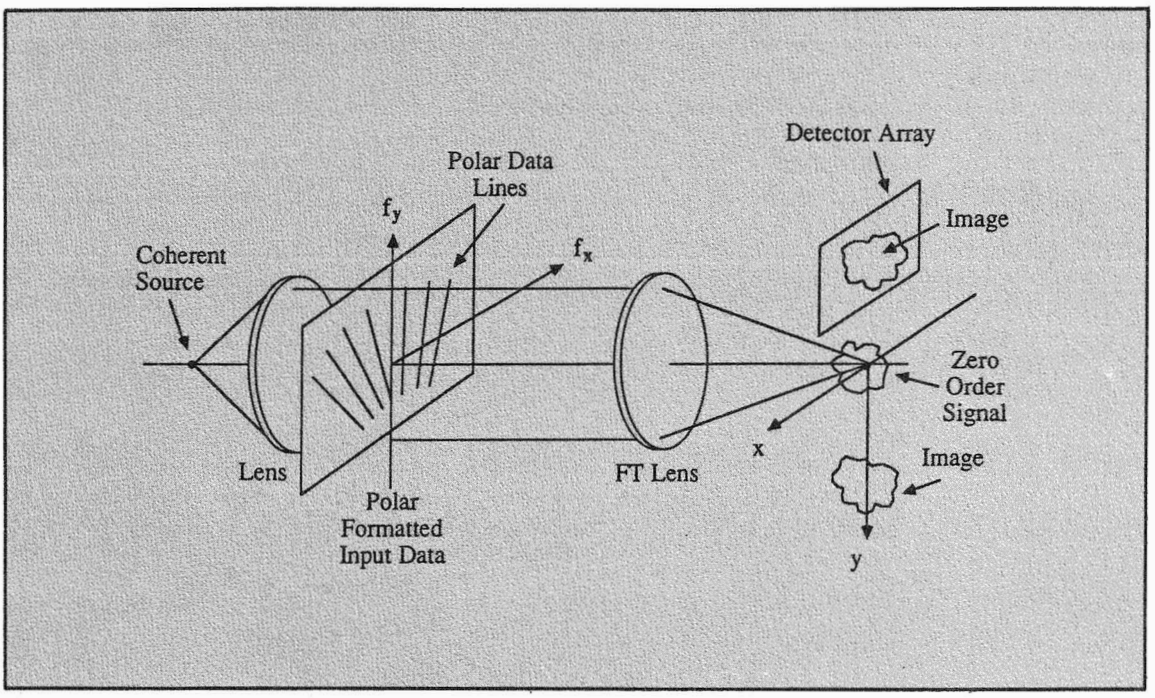
R.J. Marks II, S. Oh, L.E. Atlas and J.A. Ritcey, "Homogeneous and layered alternating projection neural networks",
in Real-Time Signal Processing for Industrial Applications, edited by Bahram Javidi
(SPIE Optical Engineering Press, Bellingham, WA. 1989), pp. 217-232.

Critical Reviews
of Optical Science
and Technology

CRITICAL REVIEWS

Real-Time Signal Processing for Industrial Applications

SPIE Volume 960



Bahram Javidi
Editor

27–28 June 1988
Dearborn, Michigan

SPIE OPTICAL ENGINEERING PRESS

Scholarly technical publications in optical and optoelectronic
applied science and engineering

HOMOGENEOUS AND LAYERED ALTERNATING PROJECTION NEURAL NETWORKS

R.J. Marks II, S. Oh, L.E. Atlas and J.A. Ritcey,
Interactive Systems Design Laboratory
University of Washington, FT-10
Seattle, WA 98195

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 alternately projecting between two or more constraint sets. Criteria for desired and unique convergence are easily established. The network can be taught from a training set by viewing each library vector only once. The network can be configured as either a content addressable memory (homogeneous form) or classifier (layered form). The number of patterns that can be stored in the network is on the order of the number of input and hidden neurons. If the output neurons can take on only one of two states, then the trained layered APNN can be easily configured to converge in one iteration. More generally, convergence is at an exponential rate. Convergence can be improved by the use of sigmoid type nonlinearities, network relaxation and/or increasing the number of neurons in the hidden layer. The manner in which the network generalizes can be directly evaluated.

1. INTRODUCTION

The current flurry of interest in artificial neural networks is motivated by recent promising research results and is predicated upon the high performance of actual biological neural networks. Artificial neural networks, which represent a departure from conventional computing techniques, have the properties of fault tolerance, resiliance to processing inexactness, a regularized structure and asynchronous operation.

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 projecting 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.

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 generally reconstruct any one library vector by clamping a subset of the neurons to the values equal to the elements of that vector. The states of the remaining floating neurons will then converge to the unknown vector elements.

The input neurons of the layered APNN provide the network's stimulus. Use of neurons in the hidden layer increases storage capacity, convergence rate and classification diversity. The states of the output layer 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. Criteria can be established for proper iterative convergence. 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 [4-7]. Such procedures generally do not address the accuracy of the final solution. In order to assure that such networks arrive at the desired globally minimum energy, computationaly lengthly procedures such as simulated annealing [8- 10] are used. For synchronous networks, steady state oscillation can occur between two states of the same energy [11]. The accuracy of the steady state solution of APNN's, on the other hand, can be straightforwardly assured for both synchronous and asynchronous networks.

(b) Many homogeneous neural networks [4,12-14] do not scale well, i.e. the storage capacity less than doubles when the number of neurons is doubled [15,16]. 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 speed of backward error propagation learning [17,18] can be painfully slow. Layered APNN's, on the other hand, can be trained on only one pass through the data. If the network memory does not saturate, new data can easily be learned without repeating previous data. Neither is the effectiveness of recall of previous data diminished. We show that in certain important applications, the APNN will recall in one iteration. Generally, the

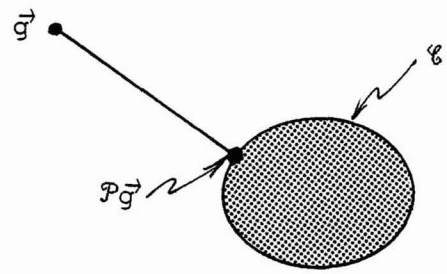


Figure 1. The two-dimensional shaded region form a convex set, $\mathcal{C}$. For an arbitrary vector $\vec{g}$, the projection onto the convex set, $P\vec{g}$, is that point in $\mathcal{C}$ closest to $\vec{g}$.

cost of this performance is more hidden neurons. Hence, we are trading computational time for architectual real estate.

(d) The manner in which the layered APNN generalizes to data for which it was not trained can be found analytically in a straightforward fashion.

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, sequential 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.

Wise use of nonlinearities is shown to improve the network's performance. Sigmoid type neural nonlinearities improve convergence properties. Establishing a hidden layer of neurons whose states are determined by a nonlinear function of the input neural states is shown to increase the network's storage capacity and increase the network's convergence rate. The manner in which the networks responds to data outside of the training set is also addressed.

2. AN OVERVIEW OF POCS

The technique of projection onto convex sets (POCS) [19,20] has traditionally been applied to signal restoration. In this section, we give a brief overview of POCS. More in depth treatment is in the book by Stark [20].

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

$$(1-\alpha)\vec{x} + \alpha\vec{y} \in \mathcal{C} \quad ; \; 0 \le \alpha \le 1$$

Geometrically, this is interpreted as requiring the line segment connecting $\vec{x}$ and $\vec{y}$ be totally subsumed in $\mathcal{C}$. The convex sets in this paper are assumed to be closed (i.e. they include their boundaries). Examples include subspaces (planes), boxes, balls and linear varieties (translated subspaces).

As illustrated in Figure 1, the projection of an arbitrary vector, $\vec{g}$, onto a convex set results in the unique vector in $\mathcal{C}$ closest to $\vec{g}$ in the mean square sense. Consider, then, I convex sets with common (convex) intersection $\mathcal{C}$:

$$\mathcal{C}_1 \cap \mathcal{C}_2 \cap \ldots \cap \mathcal{C}_I = \mathcal{C} \neq \phi$$

As is illustrated in Figure 2, the fundamental result of POCS is that repeated sequential projection onto these sets asymtotically approaches a point in $\mathcal{C}$.

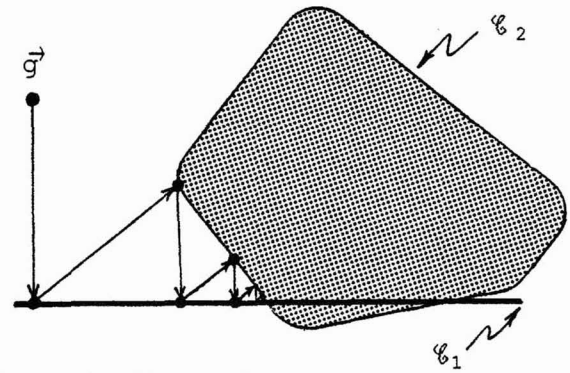


Figure 2. Illustration of POCS (projection onto convex sets). Alternating projections between the (convex) line segment, $\mathcal{C}_1$, and the shaded convex set, $\mathcal{C}_2$, asymtotically approaches a point common to their intersection.

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.

3. THE ALTERNATING PROJECTION NEURAL NETWORK

In this section, we established the notation for the APNN. Nonlinear modificiations 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 $L > N$: $\{ \vec{f}_n \mid 0 \le n \le N \}$. We form the library matrix

$$\underline{F} = [\, \vec{f}_1 \mid \vec{f}_2 \mid \ldots \mid \vec{f}_N \,]$$

and the neural network interconnect matrix

$$\underline{T} = \underline{F} \, (\underline{F}^T \underline{F})^{-1} \, \underline{F}^T \qquad (1)$$

where the superscript T denotes transposition. The interconnect value between neurons p and k is t_{pk}. Since $\underline{T}$ 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 k^{th} neuron at time M. If the k^{th} neuron falls into the known category, its state is *clamped* to the known value (i.e. $s_k(M) = f_k$ where $\vec{f}$ 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

$$i_k = \sum_{p=1}^{L} t_{pk}\ s_p \qquad (2)$$

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 extrapolated library vector if the first P rows of $\underline{F}$ (denoted $\underline{F}_P$) form a matrix of full column rank. That is, no column of $\underline{F}_P$ 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 3. Both linear and nonlinear alteration techniques to improve the network's convergence rate are in section 4.

Lastly, note that subsumed in the criterion that $\underline{F}_P$ be full rank is the condition that the number of library vectors not exceed the number of known neural states ($P \geq N$). Techniques to bypass this restriction by using hidden neurons are discussed in section 6.

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 Figure 3a. 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 Figures 3b-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

$$\underline{F}_P = [\ \vec{f}_1^P\ |\vec{f}_2^P\ |\ldots|\vec{f}_N^P\ ]$$

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

[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 [19,20] convergence imply that $\vec{s}(M) \longrightarrow \vec{f}$ as $M \longrightarrow \infty$.

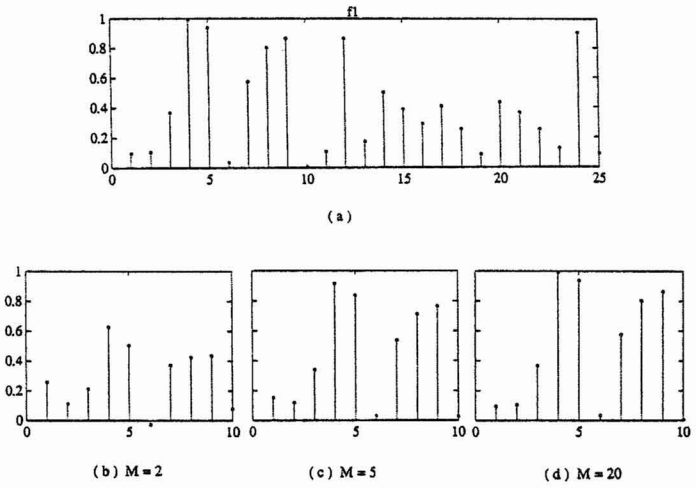


Figure 3. Illustration of homogeneous APNN results. Four library vectors of length L=23 were stochastically generated. The values of the vector $\vec{f}_1$ are shown in (a). The states of the final 15 values of $\vec{f}_1$. The states of the first ten floating neurons are shown in (b), (c) and (d) after M=2,5 and 20 synchronous iterations. The first ten elements of $\vec{f}_1$ have been clearly reconstructed in (d).

$$\underline{\eta}\,\vec{i} = \left[\begin{array}{c} \vec{f}^P \\ \hline \vec{i}_Q \end{array}\right]$$

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

$$\underline{T} = \left[\begin{array}{c|c} \underline{T}_2 & \underline{T}_1 \\ \hline \underline{T}_3 & \underline{T}_4 \end{array}\right]$$

where $\underline{T}_2$ is a P by P and $\underline{T}_4$ a Q by Q matrix. The subscripts are motivated by quadrant location. Since $\underline{T}$ is symmetric ($T = T^T$), so are $\underline{T}_2$ and $\underline{T}_4$. Furthermore $\underline{T}_1 = \underline{T}_3^T$.

4. STEADY STATE CONVERGENCE PROOFS

In this section, we prove convergence of the APNNs for both synchronous and sequential 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 $\underline{F}_P$ be full rank. The behaviour of the network when $\underline{F}_P$ is not full rank is also addressed.

Synchronous Operation

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

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

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

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

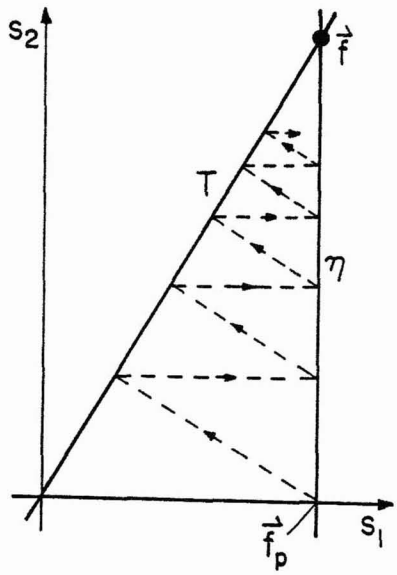


Figure 4. The linear variety, η, and the subspace T intersect at the library vector, $\vec{f}$. By alternatly projecting between the subspace and linear variety, the neural network is seen to converge to a point common to both.

Thus, the iterative state equation can be written as

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

As is illustrated in Figure 4, this operation can easily be visualized in an L dimensional signal space. The $\underline{T}$ matrix orthogonally projects any vector onto a N dimensional subspace, T, formed by the closure of the library vectors [21]. (Kohonen [22] has suggested a single projection onto T as an associative memory algorithm). The clamping operator, $\underline{\eta}$, 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 projections between these two convex sets strongly converge 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 [23,24].) Clearly, the library vector $\vec{f}$ is common to both T and η. The requirement that $\underline{F}_p$ 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

$$\left[\begin{array}{c}\vec{f}^P \\ \hline \vec{s}^Q(M+1)\end{array}\right] = \underline{\eta}\left[\begin{array}{c|c}\underline{T}_2 & \underline{T}_1 \\ \hline \underline{T}_3 & \underline{T}_4\end{array}\right]\left[\begin{array}{c}\vec{f}^P \\ \hline \vec{s}^Q(M)\end{array}\right] \tag{4}$$

The states of the P clamped neurons are not affected by their input sum. Thus, there is no contribution to the iteration by $\underline{T}_1$ and $\underline{T}_2$. We can

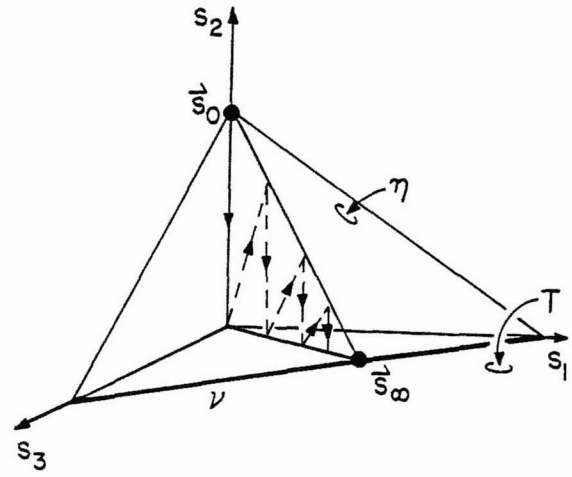


Figure 5. Illustration of the case where the subspace T and linear variety, η, intersect along the linear variety V. In such underdetermined cases, the APNN will iteratively converge to that point on V closest to the initialization. e.g. as shown, for an initialization of $\vec{s}(0)$, convergence is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

equivalently write (4) as

$$\vec{s}^Q(M+1) = \left[\begin{array}{c|c}\underline{T}_3 & \underline{T}_4\end{array}\right]\left[\begin{array}{c}\vec{f}^P \\ \hline \vec{s}^Q(M)\end{array}\right]$$

or

$$\vec{s}^Q(M+1) = \underline{T}_3\vec{f}^P + \underline{T}_4\vec{s}^Q(M) \tag{5}$$

We show in Appendix A that if $\underline{F}_p$ is full rank, then the spectral radius (magnitude of the maximum eigenvalue) of $\underline{T}_4$ is strictly less than one. It follows that the steady state solution of (5) is:

$$\vec{f}^Q = (\underline{I} - \underline{T}_4)^{-1}\,\underline{T}_3\,\vec{f}^P \tag{6}$$

where, since $\underline{F}_p$ 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

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

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 $\underline{T}_3$ and $\underline{T}_4$. Even if the partition remains static, (6) is not amenable to learning sequentially presented library vectors. We show in Section 6 that the interconnect matrix in (1) is.

Sequential Operation

An asynchronous neural network can be defined as one in which two or more neurons do not change

state at the same time [11]. Subsumed in this concept is sequential operation wherein neural states are updated periodically in an indexed order. That is, neuron 1 is allowed to change. Then neuron 2 is updated, etc. After every (floating) neuron is allowed to change, the procedure is iteratively repeated.

The convergence proof for sequential operation is based on (6) and (7) which can be written as

$$\underline{A}\,\vec{y} = \vec{b} \tag{8}$$

where $\underline{A} = (\underline{I} - \underline{T}_4)$, $\vec{b} = \underline{T}_3 \vec{\ell}^{\,P}$ and $\vec{y} = \vec{s}^{\,Q}(\infty)$. Then, the sequential operation

$$x_i(M+1) = -\sum_{k=1}^{i-1} a_{ik}\, x_k(M+1) + (1 - a_{ii})\, x_i(M) - \sum_{k=i+1}^{L} a_{ik}\, x_k(M) + b_i \tag{9}$$

converges to the desired vector, $\vec{x}(\infty) = \vec{y}$, if the spectral radius of $\underline{T}_4$ is less than one (i.e. $\underline{F}_p$ is full rank). The proof, similar to that for the Gauss-Seidel algorithm [25], is given in Appendix B.

Temporally Nondispersive Clock Skew

If we assume that the APNN reaches a stable solution, then we can establish a convergence criterion in which both synchronous and sequential operation are subsumed. Let τ_{kp} denote the time delay between floating neurons k and p and let γ_{kp} denote the time delay betwen the p^{th} clamped neuron and the k^{th} floating neuron. Then, using our partition convention, the iteration in (2) becomes

$$s_k\{t\} = \sum_{p=1}^{P} t_{kp}\ell_p^P\, \mu\{t - \gamma_{kp}\} + \sum_{p=P+1}^{L} t_{kp} s_p\{t - \tau_{kp}\}$$

for $P < k \le L$, and $\mu\{\cdot\}$ 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 $t \longrightarrow \infty$ and assuming a stable solution gives

$$s_k\{\infty\} = \sum_{p=1}^{P} t_{kp}\ell_p^P + \sum_{p=P+1}^{L} t_{kp} s_p\{\infty\} \;;\; P < k \le L$$

or, in ,matrix-vector form

$$\vec{s}^{\,Q}\{\infty\} = \underline{T}_3 \vec{\ell}^{\,P}_p + \underline{T}_4 \vec{s}^{\,Q}\{\infty\}$$

Since $\underline{T}_4$ is not singular if $\underline{F}_p$ 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 $\vec{s}^{\,Q}\{\infty\} = \vec{\ell}^{\,P}_p$.

Results of Noncompliance with Conversion Criteria

(a) *The Underdetermined Case*:

If $\underline{F}_p$ 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, $\vec{s}(0)$ [26]. Equivalently, $\vec{s}(\infty)$ is the orthogonal projection of $\vec{s}(0)$ onto $\mathcal{V}$. This result is geometrically illustrated in Figure 5.

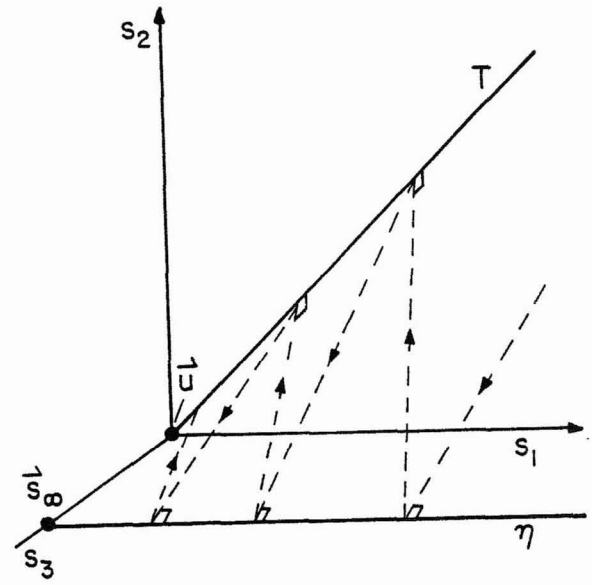


Figure 6. Two nonintersecting lines are shown in three space. the subspace T is in the (s_1, s_2) plane and the linear variety variety, η, is on the (s_1, s_3) plane. Alternating orthogonal projections between the two lines iteratively converge to a limit cycle between the two points on each line closest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

(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 $\underline{F}_p$, then $\vec{s}^{\,Q}(\infty)$ will be the same linear combination of the columns of $\underline{F}_Q$.

(b) Otherwise, the linear variety η formed by the initialization does not intersect the subspace T. As illustrated in Figure 6, the networks will converge to that point on the linear variety closest to the subspace.

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 Figure 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 the clamped neurons thus implies improper clamped values.

4. CONVERGENCE DYNAMICS

In this section, we explore different convergence dynamics of the synchronous APNN when $\underline{F}_p$ is of full column rank. If the library matrix displays certain orthogonality characteristics, or if there is a single output (floating) neuron, convergence can be achieved in a single iteration. More generally, convergence is at an exponential rate. Two techniques are presented to improve convergence. The first is standard relaxation. The

second imposes a nonlinear convex constraint at each neuron.

One Step Convergence

As can be visualized geometrically from Figure 4, if the linear variety and subspace are orthogonal, then the APNN will converge uniformly in one iteration. We show in Appendix C, however, that such one step convergence cannot occur when the values of the floating neurons in steady state are other than zero.

There remain, however, at least two important cases, where the APNN converges other than uniformly in one iteration. Both require that the output be bipolar (±1). Convergence is in one step in the sense that

$$\vec{\ell}^{Q} = \underline{\text{sign}}\ \vec{s}^{Q}(1) \tag{10}$$

where the vector operation sign takes the sign of each element of the vector on which it operates.

CASE 1:
If there is a single output neuron, then, from (5),(6) and (7)

$$s^{Q}(1) = (1 - t_{LL})\,\ell^{Q}$$

Since the eigenvalue of the (scalar) matrix, $\underline{T}_4 = t_{LL}$ lies between zero and one (see Appendix A) we conclude that $1 - t_{LL} > 0$. Thus, if ℓ^{Q} is restricted to ±1, (10) follows immediately.

A technique to extend this result to an arbitrary number of output neurons in a layered network is discussed in section 8.

CASE 2:
For certain library matrices, the APNN can also display one step convergence. We show in Appendix D, for example, that in the unlikely event that the columns of $\underline{F}$ are orthogonal and the columns of $\underline{F}_p$ are also orthogonal, then one synchronous iteration results in floating states proportional to the steady state values. Specifically, for the floating neurons,

$$\vec{s}^{Q}(1) = \frac{\|\vec{\ell}^{p}\|^{2}}{\|\vec{\ell}\|^{2}}\,\vec{\ell}^{Q} \tag{11}$$

Exponential Convergence

More generally, the convergence rate of the APNN is exponential and is a function of the eigenstructure of $\underline{T}_4$. Let $\{\vec{p}_r \mid 1 \le r \le Q\}$ denote the eigenvectors of $\underline{T}_4$ and $\{\lambda_r\}$ the corresponding eigenvalues. Define

$$\underline{P} = [\,\vec{p}_1 \mid \vec{p}_2 \mid \ldots \mid \vec{p}_Q\,]$$

and the diagonal matrix $\underline{\Lambda}_4$ such that

$$\text{diag}\ \underline{\Lambda}_4 = [\lambda_1\ \lambda_2\ \ldots \lambda_Q]^{T}$$

Then we can write

$$\underline{T}_4 = \underline{P}\,\underline{\Lambda}_4\,\underline{P}^{T}$$

Define

$$\vec{x}(M) = \underline{P}^{T}\vec{s}(M)$$

Since $\underline{P}\,\underline{P}^{T} = \underline{I}$, it follows from the difference equation in (5) that

$$\vec{x}(M+1) = \underline{P}^{T}\underline{T}_4\,\underline{P}\,\underline{P}^{T}\vec{s}(M) + \underline{P}^{T}\underline{T}_3\vec{\ell}^{p}$$

$$= \underline{\Lambda}_4\,\vec{x}(M) + \vec{g}$$

where

$$\vec{g} = \underline{P}^{T}\underline{T}_3\vec{\ell}^{p}$$

The solution to this difference equation is

$$x_k(M) = \sum_{r=0}^{M} \lambda_k^{r}\, g_k$$

$$= [\,1 - \lambda_k^{M+1}\,]\,(1 - \lambda_k)^{-1}\, g_k \tag{12}$$

In appendix A, we show that the spectral radius of $\underline{T}_4$ is less than one. Thus $\lambda_k^{M} \to 0$ as $M \to \infty$. Our steady state result is thus

$$x_k(\infty) = (1 - \lambda_k)^{-1}\, g_k$$

Equation (12) can therefore be written as

$$x_k(M) = [\,1 - \lambda_k^{M+1}\,]\, x_k(\infty)$$

The equivalent of a "time constant" in this exponential convergence is $1/\ell n(1/|\lambda_k|)$. The speed of convergence is thus dictated by the spectral radius of $\underline{T}_4$. As we will show later, adding neurons in a hidden layer in an APNN can significiantly reduce this spectral radius and thus improve the convergence rate.

Relaxation

Both the projection and clamping operations can be relaxed to alter the network's convergence without affecting its steady state [19]. For the interconnects, we choose an appropriate value of the relaxation parameter θ in the interval (0,2) and redefine the interconnect matrix as

$$\underline{T}^{\theta} = \theta\underline{T} + (1 - \theta)\underline{I}$$

or equivalently,

$$t_{nm}^{\theta} = \begin{cases} \theta\,(t_{nn} - 1) + 1 & ;\ n = m \\ \theta\, t_{nm} & ;\ n \ne m \end{cases}$$

To see the effect of such relaxation on convergence, we need simply examine the resulting eigenvalues. If $\underline{T}_4$ has eigenvalues $\{\lambda_r\}$, then $\underline{T}_4^{\theta}$ has eigenvalues

$$\lambda_r^{\theta} = 1 + \theta\,(\lambda_r - 1)$$

A wise choice of θ reduces the spectral radius of $\underline{T}_4^{\theta}$ with respect to that of $\underline{T}_4$ and thus decreases the time constant of the network's convergence. We show in Appendix E that, for synchronous operation, a "good" stationary choice for θ is

$$\theta = \text{tr}(\underline{I} - \underline{T}_4)/\text{tr}[\,(\underline{I} - \underline{T}_4)^{2}\,] \tag{13}$$

where tr denotes the matrix trace. Varying relaxation with each iteration is also a possibility [29]. Interconnect weights, however, would need to be updated each iteration.

Any of the operators projecting onto convex sets can be relaxed without affecting steady state

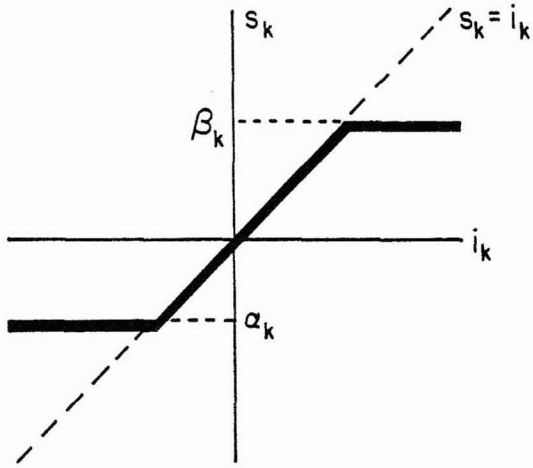


Figure 7. A saturation nonlinearity. The state of the k^{th} neuron, s_k, is determined by the sum of the inputs to that neuron, i_k.

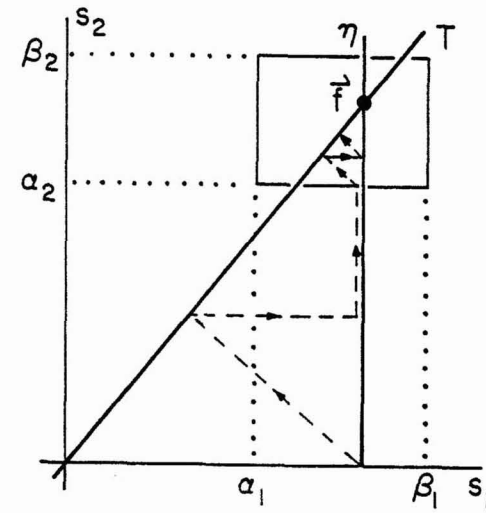


Figure 8. A geometrical illustration of the effect of nonlinear neural saturation shown in Figure 4. In addition to alternately projecting between the subspace, T, and the linear variety, η, projection is also onto a (convex) box the dimensions of which are determined by the neural saturation parameters.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

convergence [19-20]. These include the η operator and, as discussed in the section to follow, the operator that projects onto a box. Choice of stationary relaxation parameters without numerical and/or empirical study of each specific case, however, generally remains more of an art than a science.

Neural Saturation as a Convex Constraint

An alternate technique for improving the convergence rate is by imposing additional convex constraints in the iteration process. Consider, for example, placing a dynamic range constraint on each floating node:

$$s_k = \begin{cases} \alpha_k & ; \ i_k \le \alpha_k \\ i_k & ; \ \alpha_k \le i_k \le \beta_k \\ \beta_k & ; \ i_k \ge \beta_k \end{cases}$$

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 Figure 7 is similar in form to sigmoid nonlinearities used in other neural networks [18].

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 k^{th} element of a new vector lies between α_k and β_k, then no change is 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

$$\alpha_k = \min_{1 \le n \le N} \ell_n(k)$$

and

$$\beta_k = \max_{1 \le n \le N} \ell_n(k)$$

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 Figure 8, the convergence of the net can be improved by this procedure. Convergence follows immediately from POCS for I=3 convex sets.

6. TRAINING

The equation for the interconnect matrix in (1) is 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, $\underline{T}$, and wish to update the interconnects corresponding to a new library vector, $\vec{\ell}$. As illustrated in Figure 9, $\underline{T}\vec{\ell}$ projects $\vec{\ell}$ onto T and

$$\vec{\varepsilon} = (\underline{I} - \underline{T})\vec{\ell}$$

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

We wish to extend the dimension of the subspace T by one in the direction of $\vec{\varepsilon}$. Since $\vec{\varepsilon}/\|\vec{\varepsilon}\|$ is the unit vector orthogonal to T, the updated interconnect matrix

$$\underline{T}^+ = \underline{T} + \frac{\vec{\varepsilon}\,\vec{\varepsilon}^T}{\vec{\varepsilon}^T\vec{\varepsilon}}$$

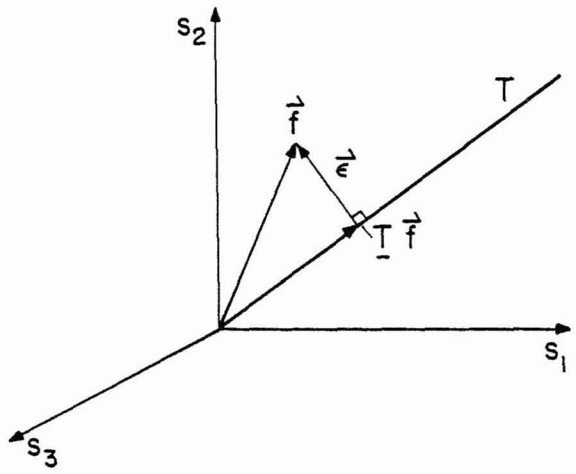


Figure 9. A geometrical illustration of learning in an APNN. The subspace, T, shown as a line on the (s_1,s_2) plane, is to be augmented to include the new library vector, $\vec{f}$, which also lies in the (s_1,s_2) plane. The error vector, $\vec{\epsilon}$, is determined by the old networks' interconnects. 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.

---

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 $(\underline{I} - \underline{T})\vec{f} = \vec{0}$, 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 $\vec{\epsilon}^T\vec{\epsilon}$ to some appropriate threshold prior to updating [30].

7. 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 by Marks [1].

Hidden Layers

In its generic form, the APNN cannot perform a simple two bit parity check. Indeed, failure to perform this same operation was a nail in the coffin of the perceptron [31]. Rumelhart *et. al.* [17,18] revived the perceptron by adding additional layers of neurons, thereby allowing nonlinear discrimination. With the addition of a hidden layer, the APNN likewise generalizes.

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 a two bit parity check is

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

The APNN cannot be used to faithfully execute this operation since

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

is not full column rank. Our approach is to augment $\underline{F}_P$ 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 [31- 36] and, in addition to the layered perceptron, is used in Φ -classifiers [32] and potential function classifiers [34]. 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 neurons 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

$$\underline{F}_+ = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ \hline 1 & e & e & e^2 \\ 1 & 0 & 0 & -1 \\ \hline 0 & 1 & 1 & 0 \end{bmatrix} \qquad (14)$$

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 Figure 10. The interconnect values for the dashed lines are unity. The remaining interconnects are from the projection matrix formed from $\underline{F}_+$.

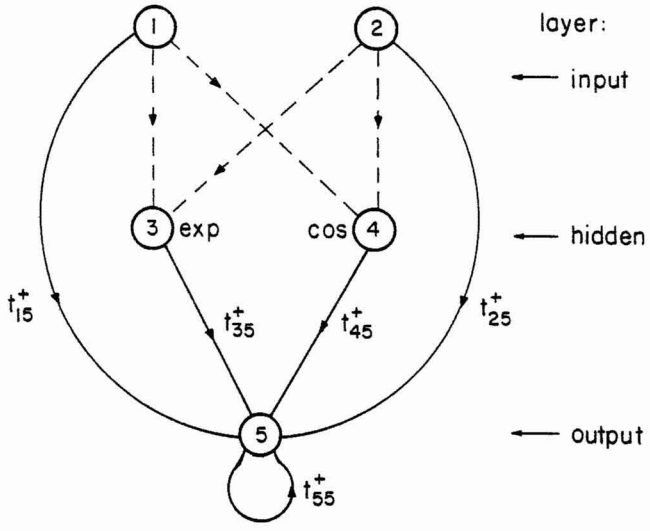


Figure 10. Illustration of a layered APNN for performing parity checking. When the states of the input layer are clamped, the hidden layer units are, as a result, also clamped. The transmittance of the dashed line interconnects, in this example, are all unity. The remainder are from the projection matrix vaues using $\underline{F}_+$ in (13) as the library matrix. Numerically, $t_{35} = -t_{45} = -0.0375$, $t_{15} = t_{25} = 0.1573$, $t_{55} = 0.9446$. In ten synchronous iterations, clamping the inputs to either (0,0) or (1,1) results in an output of 0.00 wheras clamping either to (0,1) or (1,0) results in 0.996.

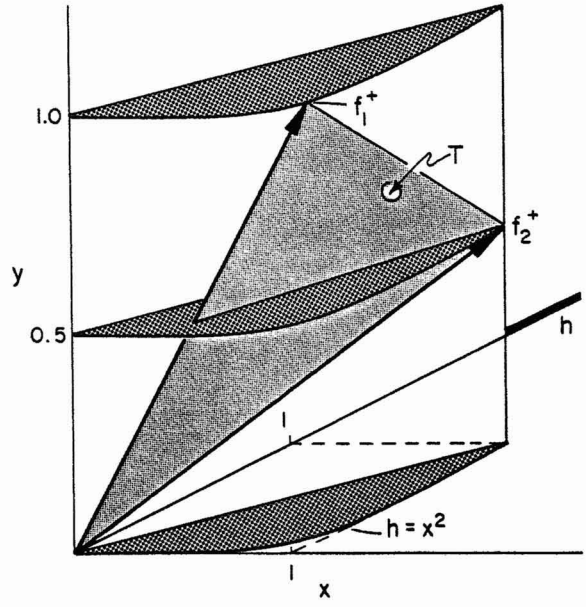


Figure 11. A geometrical illustration of the use of an x^2 nonlinearity to determine the states of hidden neurons. The library vector $\vec{f}_1^+$ is the only point in the subspace, quadratic surface and the linear variety $x = 1/2$. For $\vec{f}_2^+$, the linear variety corresponding to the clamped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

Geometrical Interpretation

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

$$\underline{F} = \begin{bmatrix} 1/2 & 1 \\ 1 & 1/2 \end{bmatrix}$$

Clearly $\underline{F}_p = [1/2\ 1]$. Let the neurons in the hidden layer be determined by the nonlineariy x^2 where x denotes the elements in the first row of $\underline{F}$. Then

$$\underline{F}_+ = \left[\vec{f}_1^+ \mid \vec{f}_2^+ \right] = \begin{bmatrix} 1/2 & 1 \\ 1/4 & 1 \\ 1 & 1/2 \end{bmatrix} \qquad (15)$$

The corresponding geometry is shown in Figure 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 suface 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 $\underline{T}_4$ matrix is decreased as additional neurons are added. The dominant time constant controlling convergence is thus increased. A proof is in Appendix F.

Capacity

Under the assumption that nonlinearities are chosen such that the augmented $\underline{F}_p$ 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.

7. 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 check problem. Let $s_5(M)$ denote the state of the output neuron at the M^{th} (synchronous) iteration. If s_1 and s_2 denote the input clamped value, then

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

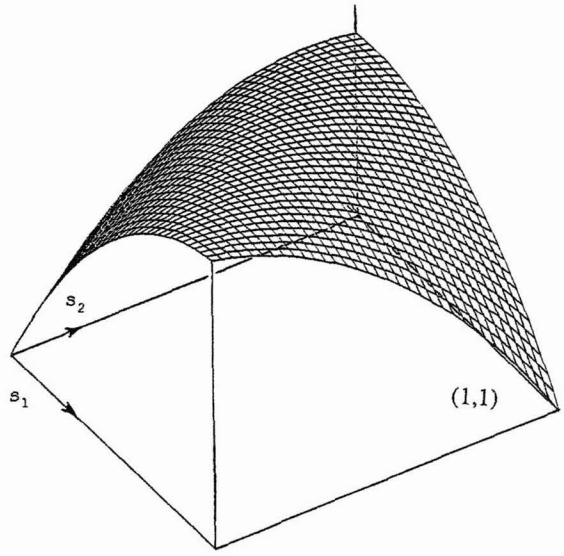


Figure 12. Response of the elementary parity check APNN using an exponential and trignometric nonlinearity in the hidden layer. Note that, at the corners, the function is equal to the XOR of the coordinates.

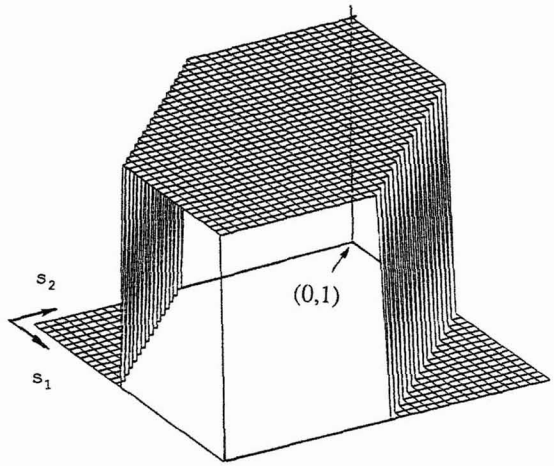


Figure 13. The generalization of the parity check APNN networks formed by thresholding the function in Figure 12 at 3/4. Different hidden layer nonlinearities result in different generalizations.

where

$$s_3 = \exp(s_1 + s_2) \tag{16}$$

and

$$s_4 = \cos\frac{\pi}{2}(s_1 + s_2) \tag{17}$$

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

$$s_5(\infty) = \frac{1}{1 - t_{55}} [t_{15}s_1 + t_{25}s_2 + t_{35}\exp(s_1+s_2) + t_{45}\cos\frac{\pi}{2}(s_1+s_2)] \tag{18}$$

A plot of $s_5(\infty)$ versus (s_1, s_2) is shown in Figure 12. The plot clearly goes through 1 and zero according to the parity of the corner coordinates. Thresholding Figure 12 at 3/4 results in the generalization perspective plot shown in Figure 13.

Note that the equapotential contours in Figure 12 are all parallel to line $s_1 + s_2 = 0$. This is because the nonlinearities in (16) and (17) are both a function of $s_1 + s_2$ and therefore have the same equapotential contours as $s_1 + s_2 = 0$. Since $t_{15} = t_{25} = -0.2471$, (18) is strictly a function of $s_1 + s_2$. The generalization is therfore dictated by our choice of nonlinearities.

Greater flexibility in classification can be achieved by training the nonlinearities [17-18] 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 procide the stimulus for the floating output neurons. We continue with the parity check example, except redefine our library matrix with -1 denoting a logic zero:

$$\underline{F} = \begin{bmatrix} -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{bmatrix}$$

One advantage of this redefinition is the obvious choice of zero as an output threshold [38].

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 Figure 14 was obtained by thresholding the output at zero and is nearly a least mean square partition. 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 [37] from a distribution uniform on $(-\frac{1}{2}, \frac{1}{2})$. The nonlinearity was exp(-z). Generalizations are shown in Figure 15 for 10 and 50 hidden neurons. The generalization here also approaches a least mean square partition.

9. NOTES

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

(b) Nonlinearities can also be used to increase the capacity of the homogeneous APNN. Invision, for example, associating with each neuron a single hidden neuron whose

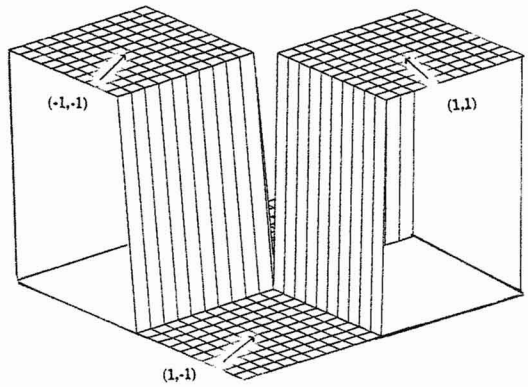


Figure 14. Generalization of the parity check APNN using 4 hidden neurons and the nonlinearity exp(-z) for each hidden neuron.

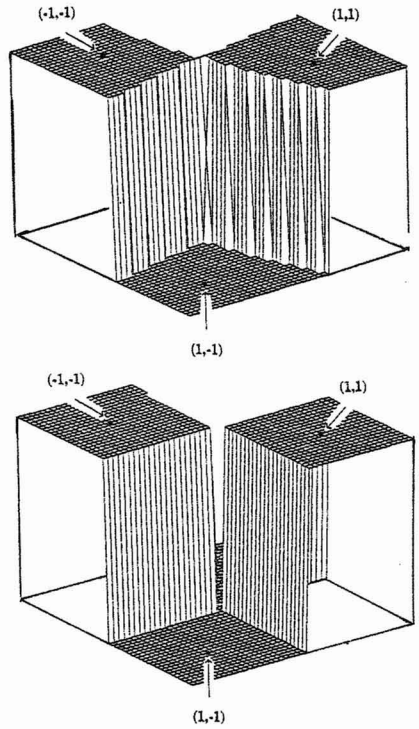


Figure 15. Generalization of the parity check APNN using 10 (top) and 50 (bottom) hidden neurons and stochastic interconnects from the input to the hidden layer.

state is related is some fixed nonlinear manner. Clamping P neurons then clamps an additional P hidden neurons and the network's capacity is essentially doubled.

(c) 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.

(d) When teaching a layered APNN, the error of only the output neurons should be used to determine whether the interconnects need to be updated. After one pass through the data, inputs which were not used to update the interconnects should be rechecked.

(e) 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.

(f) There is a possibility that for a given set of hidden neuron nonlinearities, augmentation of the $\underline{F}_p$ matrix coincidentally will 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.

(g) We have shown in section 4 that if an APNN has a single bipolar output neuron, then the network converges in one step in the sense of (10). Visualize a layered APNN with a single output neuron. If there are a sufficiently large number of neurons in the hidden layer, then the input layer does not need to be connected to the output layer. That is, the input layer clamps the hidden layer which alone acts as the output neuron's stimulus. Consider a second neural network identical to the first in the input and hidden layers. The hidden to output interconnects are chosen, however, to generate a different bipolar output. Since the top two layers and their interconnects are identical in both cases, the two networks can be combined into a third composite network identical in the first two layers, but with two output neurons. The interconnects from the hidden layer to the output neurons are identical to those used in the single output neurons architectures. Since the component networks converge in a single step, the composite network will also. The process can clearly be extended to an arbitrary number of output neurons.

(h) Recently, optical architectures have been proposed for implementing the APNN [39-40]. Iteration is performed at light speed.

Acknowledgements

The author express their appreciation to D.C. Park and Dr. Kwan F. Cheung for many insightful discussions. This research was supported by the Washington Technology Center and SDIO/IST's ultrahigh speed computing program administered through the U.S. Offices of Naval Research in conjunction with the Optical Systems Laboratory at Texas Tech University. L.E. Atlas was also partially supported by a National Science Foundation Presidential Young Investigator Award and J.A. Ritcey by a Physio-Control career development grant.

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," Proc. IEEE 1st International Conf. on Neural Networks, San Diego, 1987.

3. R.J. Marks II, L.E. Atlas and K.F. Cheung, "A Class of Continuous Level Neural Nets," Proc. 14th Congress of International Commission for Optics Conf., Quebec, Canada, 1987.

4. J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," Proceedings of the National Academy of Sciences, USA, vol.79, pp.2554-2558, 1982.

5. J.J. Hopfield and D.W. Tank, "Neural Computation of Decisions in Optimization Problem," Biological Cybernatics, vol.52, pp.141-152, 1985.

6. D.W. Tank and J.J.Hopfield, "Simple Neurel Optimization Networks: an A/D Converter, Signal Decision Circuit and a Linear Programming Circuit," IEEE Trans. Circuits and Systems, 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. Pattern Recognition and Machine Intelligence, vol. PAMI-6, pp.721-741, 1984.

9. S. Kirkpatrick, C.D. Gelatt Jr. and M.P. Vecchi, "Optimization by Simulated Annealing," Science, vol. 220, no. 4598, pp. 671-680, 1983.

10. D.H. Ackley, G.E. Hinton, and T.J. Senjnowski, "A Learning Algorithm for Boltzmann Machines," Cognitive Science, 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," to appear in Applied Optics.

12. R.P. Lippman, "An Introduction to Computing With Neural nets," IEEE ASSP Magazine, p.7, April 1987.

13. N. Farhat, D. Psaltis and E. Paek, "Optical Implementation of the Neural Model," Appl. Opt., vol. 24, pp.1469, 1985.

14. L.E. Atlas, "Auditory Coding in Higher Centers of the CNS, "IEEE Eng. in Medicine and Biology Magazine, pp. 29-32, June 1987.

15. Y.S. Abu-Mostafa and J.M. St. Jaques, "Information Capacity of the Hopfield Model, " IEEE Trans. Inf. Theory, vol. IT-31, p.461, 1985.

16. R.J. McEliece, E.C. Posner, E.R. Rodemich and S.S. Venkatesh, " The Capacity of the Hopfield Associative Memory, " IEEE Trans. Inf. Theory (submitted), 1986.

17. D.E. Rumelhart, J.L. McClelland and the PDP Research Group, **Parallel Distributed Processing**, vol. I and II, Bradford Books, Cambridge, MA, 1986.

18. 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.

19. D.C. Youla & H. Webb, "Image Restoration by the Method of Convex Projections: Part I-Theory," IEEE Trans. Med. Imaging, vol. MI-1, pp.81- 94, 1982 and M.I. Sezan & 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.

20. H. Stark. **Image Recovery**, Academic Press, New York (1987).

21. G. Strang, **Linear Algebra and Its Applications**, Academic, New York, 1980, p.116.

22. T. Kohonen, **Self-Organization and Associative Memory**, Second Edition, Springer- Verlag, Berlin, 1988.

23. J. Von Neumann, **The Geometry of Orthogonal Spaces**, Princeton U. Press, Princeton, N.J. 1950.

24. D.W. Montogomery, "Optical applications of von Neumann's alternating- projection theorem," Opt. Lett., vol. 7, no.1, pp.1-3, Jan. 1982.

25. A. Ralston, P. Rabinowitz, "A First Course in Numerical Analysis," 2nd Edition, McGraw-Hill, New-York, 1985.

26. M.R. Civanlar and H.J. Trussel, "Digital Signal Restoration Using Fuzzy Sets," IEEE Tran. ASSP, vol. ASSP-34, p.919, 1986.

27. M. Goldburg and R.J. Marks II, "Signal Synthesis in the Presence of an Inconsistent Set of Constraints," IEEE Trans. Circuits and Systems, vol. CAS-32, pp. 647-663, 1985.

28. D.C. Youla and V. Velasco, "Extensions of a Result on the Synthesis of Signals in the Presence of Inconsistent Constraints," IEEE Trans. Circuits and Systems, vol. CAS-33, pp. 465-468, 1986.

29. V.T. Tom, T.J. Quatieri, M.H. Hayes and J.H. McClellan, "Convergence of Iterative Nonexpansive Signal Reconstruction Algorithms," IEEE Trans. ASSP, vol. ASSP-29, no.5,p.1052, Oct. 1981.

30. 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.

31. M. Minsky and S. Papert, **Perceptrons**, MIT Press, Cambridge, MA, 1969.

32. J. Sklansky and G.N. Wassel, **Pattern Classifiers and Trainable Machines**, Springer-Verlag, New York, 1981.

33. B. Widrow and M.E. Hoff, "Adaptive Switching Circuits," 1960 **IRE WESCON Convention Record**, New York: IRE, pp. 96-104 (1960).

34. J.T. Tou and R.C. Gonsalez, **Pattern Recognition Principles**, Addision- Wesley Publishing Company, Reading, Mass, 1974.

35. F. Rosenblatt, **Principles of Neurodynamics**, Sparton Books, Washington D.C., 1962.

36. K. Fukunaga, **Introduction to statistical Pattern Recognition**, Academic Press, New York, 1972.

37. R.J. Marks II, L.E. Atlas, D.C. Park and S.Oh, "The Effect of Stochastic Interconnects in Artificial Neural Network Classification," Proc. IEEE International Conference on Neural Network, San Diego, July 24-27, 1988.

38. R.J. Marks II, J.A. Ritcey, L.E. Atlas and K.F. Cheung, "Composite Matched Filter Output Partitioning," Applied Optics, 26, pp.2274-2278 (1987).

39. S. Oh, L.E. Atlas, R.J. Marks II and D.C. Park, "Effects of Clock Skew in Iteration Neural Network and Optical Feedback Processors," Proc. IEEE International Conference on Neural Networks, San Diego, July 24-27, 1988.

40. R.J. Marks II, L.E. Atlas, S. Oh and K.F. Cheung, "Optical Processor Architectures for Alternating Projection Neural Networks," Optics Letters, 13, pp. 533-535 (1988).

APPENDIX A: *Some Properties of $\underline{T}$ and $\underline{T}_4$*

Properties of the T matrix

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

a. T is symmetric and idempotent
b. Eigenvalues of $\underline{T}$ are zero or one
c. Diagonal elements of $\underline{T}$ lie between zero and one inclusive
d. $\mathrm{tr}(\underline{T}) = N$

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

$\#(\underline{A})$ = number of eigenvalues of $\underline{A}$ equal to one
$\nu(\underline{A})$ = nullity of $\underline{A}$
$\mathrm{rank}(\underline{A}) = k - \nu(\underline{A})$

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

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

$$\underline{B} = \underline{Q}_L\,\underline{V}\,\underline{Q}_R$$

Also, $\underline{V}$ can be partitioned as:

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

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

$$\underline{B}^T\underline{B} = \underline{Q}_R\,\underline{V}^T\underline{V}\,\underline{Q}_R$$

and

$$\underline{B}\,\underline{B}^T = \underline{Q}_L\underline{V}\,\underline{V}^T\underline{Q}_L$$

Here

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

is a k by k matrix and

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

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

Lemma 2: All eigenvalues, $\{\lambda_i \mid 1 \le i \le Q\}$, of $\underline{T}_4$ are real and lie in the interval [0,1].

Proof:
Let $\underline{Q}$ denote an orthogonal matrix with the property that

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

where $\underline{I}$ is the N by N identity matrix. We partition $\underline{Q}$ as:

$$\underline{Q} = \left[\begin{array}{c|c} \underline{Q}_2 & \underline{Q}_1 \\ \hline \underline{Q}_3 & \underline{Q}_4 \end{array}\right]$$

where $\underline{Q}_2$ has dimension N by P. The matrix $\underline{T}_4$ can then be written as

$$\underline{T}_4 = \underline{Q}_1{}^T\underline{Q}_1$$

Let $\underline{S}_4 = \underline{Q}_4{}^T\underline{Q}_4$. Then $\underline{T}_4$ and $\underline{S}_4$ are symmetric matrix and $\underline{T}_4 + \underline{S}_4 = \underline{I}$. Clearly

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

where λ_i and λ_i^s are the eigenvalues of $\underline{T}_4$ and $\underline{S}_4$ respecively. Because $\underline{T}_4$ and $\underline{S}_4$ are positive semidefinite matrices,

$$\lambda_i \ge 0 \quad \text{and} \quad \lambda_i^s \ge 0$$

The proof follows immediately from these last two equations.

Lemma 3: rank $(\underline{F}_P)$ = rank $(\underline{Q}_2)$

Proof:

Consider the matrix $\underline{G} = \underline{Q}\,\underline{F}$. We write $\underline{G}$ as

$$\underline{G} = \begin{bmatrix} \underline{G}_N \\ \hline \underline{G}_M \end{bmatrix} = \underline{Q}\,\underline{F} = \underline{Q} \begin{bmatrix} \underline{F}_P \\ \hline \underline{F}_Q \end{bmatrix} \tag{A1}$$

where $\underline{G}_N$ denotes the first N rows of $\underline{G}$ and we have used the identity

$$[\underline{G}(\underline{G}^T\underline{G})^{-1}\underline{G}^T]\underline{G} = \underline{G}$$

Thus

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

or, equivalently

$$\underline{G}_M = \underline{0} \tag{A2}$$

We will show that, as a consequence, $\underline{G}_N$ is nonsingular. Clearly,

$$\det \underline{G}^T\underline{G} = \det \underline{F}^T\underline{Q}^T\underline{Q}\,\underline{F} \neq 0$$

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

$$\det \underline{G}^T\underline{G} = \det \underline{G}_N{}^T\underline{G}_N = (\det \underline{G}_N)^2 \neq 0$$

and $\underline{G}_N$ is not singular.

From equation (A1), $\underline{F}_P$ can be written as

$$\underline{F}_P = \underline{Q}_2{}^T\,\underline{G}_N$$

Thus

$$\text{rank } (\underline{F}_P) = \text{rank } (\underline{Q}_2{}^T) = \text{rank } (\underline{Q}_2)$$

and our proof is complete.

Theorem 1: $\nu(\underline{F}_P) = \#(\underline{T}_4) = \nu(\underline{I} - \underline{T}_4)$. Thus, if $\underline{F}_P$ is full rank, then the eigenvalues of $\underline{T}_4$ are strictly less than one.

Proof:

$$\nu(\underline{Q}_2\underline{Q}_2{}^T) = N - \text{rank}(\underline{Q}_2\underline{Q}_2{}^T) = N - \text{rank}(\underline{Q}_2)$$

From Lemma 3,

$$\nu(\underline{Q}_2\underline{Q}_2{}^T) = N - \text{rank}(\underline{F}_P) = \nu(\underline{F}_P)$$

Since $\underline{Q}_1\underline{Q}_1{}^T + \underline{Q}_2\underline{Q}_2{}^T = \underline{I}$, and $\underline{Q}_1\underline{Q}_1{}^T$ and $\underline{Q}_2\underline{Q}_2{}^T$ are symmetric, we conclude from Lemma 1 and Lemma 2 that:

$$\#(\underline{T}_4) = \#(\underline{Q}_1\underline{Q}_1{}^T) = \nu(\underline{Q}_2\underline{Q}_2{}^T) = \nu(\underline{F}_P)$$

and

$$\#(\underline{T}_4) = \nu(\underline{F}_P)$$

APPENDIX B: *Proof of Sequential Convergence*

If the spectral radius of $\underline{T}_4$ is less than one, then the sequential operation in (9) converges to the desired vector

$$\vec{y} = (\underline{I} - \underline{T}_4)^{-1}\,\underline{T}_3\,\vec{f}^P$$

Proof:

Since $\underline{A}$ is a symmetric matrix, $\underline{A}$ can be written as

$$\underline{A} = \underline{L} + \underline{I} + \underline{D} + \underline{L}^T$$

where $\underline{L}$ is a lower triangular matrix with a zero diagonal and $\underline{D}$ is a diagonal matrix. The sequential operation in (9) can then be written in matrix form as

$$\vec{x}(M+1) = -\underline{L}\,\vec{x}(M+1) - (\underline{D} + \underline{L}^T)\vec{x}(M) + \vec{b} \tag{B1}$$

or equivalently, in state equation form:

$$\vec{x}(M+1) = -(\underline{I} + \underline{L})^{-1}(\underline{D} + \underline{L}^T)\vec{x}(M) + (\underline{I} + \underline{L})^{-1}\vec{b}$$

If the spectral radius of $(\underline{I} + \underline{L})^{-1}(\underline{D} + \underline{L}^T)$ is smaller than one, equation (B1) will converge. Consider the eigenvalue equation corresponding to

$$(\underline{I} + \underline{L})^{-1}(\underline{D} + \underline{L}^T)\,\vec{v} = \lambda\,\vec{v}$$

Premultiplying by $\vec{v}^*(\underline{I} + \underline{L})$ gives

$$\vec{v}^*(\underline{D} + \underline{L}^T)\vec{v} = \lambda\,\vec{v}^*(\underline{I} + \underline{L})\,\vec{v} \tag{B2}$$

Adding $\vec{v}^*(\underline{I} + \underline{L})\vec{v}$ to both sides gives

$$\begin{aligned} \vec{v}^*\underline{A}\,\vec{v} &= (1 + \lambda)\,\vec{v}^*(\underline{I} + \underline{L})\,\vec{v} \\ &= (1 + \lambda)\vec{v}^*(\underline{I} + \underline{L}^T)\,\vec{v} \end{aligned} \tag{B3}$$

where we have recognized that, since $\underline{A}$ is real and symmetric, $\vec{v}^*\underline{A}\,\vec{v}$ is its own conjugate transpose. Furthermore,

$$\begin{aligned} \vec{v}^*\underline{A}\,\vec{v} &= (1 + \lambda)[\,\vec{v}^*\vec{v} + \vec{v}^*\underline{L}\,\vec{v}\,] \\ &= (1 + \lambda)[\,\vec{v}^*\vec{v} + \lambda^*\vec{v}^*(\underline{I} + \underline{L}^T)\,\vec{v} - \vec{v}^*\underline{D}\,\vec{v}\,] \end{aligned}$$

where we have substituted for $\vec{v}^*\underline{L}\,\vec{v}$ from the conjugate transpose of (B2). Equating with (B3) and rearranging gives

$$(1 - |\lambda|^2)\vec{v}^*(I + L)^T\vec{v} = (1 + \lambda)[\,\vec{v}^*(I - \underline{D})\,\vec{v}\,] \tag{B4}$$

From (B3) and (B4):

$$(1 - |\lambda|^2)\vec{v}^*\underline{A}\,\vec{v} = |1 + \lambda|^2\,\vec{v}^*(\underline{I} - \underline{D})\,\vec{v} \tag{B5}$$

Here, $A = \underline{I} - \underline{T}_4$, is positive definite and $\underline{I} - \underline{A} = -\underline{D} - \underline{L} - \underline{L}^T = \underline{T}_4$ is positive semidefinite. Furthermore, the diagonal elements of $\underline{I} - \underline{A}$ are non- negative. Thus, $\underline{I} - \underline{D}$, is positive definite. We therefore conclude from (B5) that

$$1 - |\lambda|^2 > 0$$

which implies $|\lambda| < 1$. Equation (B1) thus converges with the following limit value

$$\vec{y} = -\underline{L}\,\vec{y} - (\underline{D} + \underline{L}^T)\vec{y} + \vec{b}$$

or

$$\vec{y} = (\underline{I} - \underline{T}_4)^{-1}\,\vec{b}$$

and our proof is completed.

APPENDIX C: *Proof of Nonexistence of One Step Uniform Convergence*

Theorem 2: There is no APNN with iteration convergence to $\vec{s}(1)=\vec{s}(\infty)$ if $\vec{s}^{Q}(\infty)\neq\vec{0}$.

Proof: (Note - the notation used here was established in Appendix A).

case 1:

If $\underline{T}_4 = \underline{0}$, then $\underline{Q}_1 = \underline{0}$. If $\underline{Q}_1 = \underline{0}$, we have

$$\underline{T}_3 = \underline{Q}_1^{T}\underline{Q}_2 = \underline{0}$$

so that

$$\vec{s}(\infty) = \vec{s}(1) = \vec{0}.$$

case 2

$$\underline{T}_4 \neq \vec{0}$$

If convergence occurs in one step, then

$$\vec{s}(2) = \vec{s}(1)$$

or, equivalently

$$\underline{T}_4\underline{T}_3\vec{f}^{P} = \vec{0}$$

Consider the spectral decomposition matrices of $\underline{T}_4$, $\{\underline{E}_i = \vec{\lambda}_i\vec{\lambda}_i^{T} \mid 1\leq i\leq t\}$ such that

$$\sum_{i=1}^{t} \underline{E}_i = \underline{I}$$

and

$$\sum_{i=1}^{t} \lambda_i \underline{E}_i = \underline{T}_4$$

where t is equal to the number of distinct eigenvalues of $\underline{T}_4$. Then

$$\sum_{i=1}^{t} \lambda_i \underline{E}_i \underline{T}_3 \vec{f}^{P} = \vec{0}$$

and

$$\underline{E}_i \underline{T}_3 \vec{f}^{P} = \vec{0} \quad ; 1\leq i\leq t$$

Thus

$$\underline{T}_3 \vec{f}^{P} = \sum_{i=1}^{t} \underline{E}_i \underline{T}_3 \vec{f}^{P} = 0$$

and

$$\vec{s}(1) = \vec{s}(\infty) = \underline{T}_4\vec{s}(0) + \underline{T}_3\vec{f}^{P} = \vec{0}$$

The steady state value must then be identically zero.

APPENDIX D: *One Step Sign Convergence for an Orthogonal Library*

Here we show that if all the columns of $\underline{F}$ are orthogonal, the columns of $\underline{F}_P$ are orthogonal, and $\vec{f}^{P}$ is a column vector of $\underline{F}_P$, then

$$\vec{s}(1) = \frac{\|\vec{f}^{P}\|^2}{\|\vec{f}\|^2}\vec{f}^{Q}$$

Proof:

Since the columns of $\underline{F}$ are orthgonal, $\underline{F}^{T}\underline{F}$ is a diagonal matrix, and the i^{th} diagonal element is $\|\vec{f}_i\|^2$. From equation (1), $\underline{T}_3$ can be written as

$$\underline{T}_3 = \underline{F}_Q(\underline{F}^{T}\underline{F})^{-1}\underline{F}_P^{T}$$

If $\vec{f}^{P}$ is the i^{th} column of $\underline{F}_P$, then

$$\vec{s}(1) = \underline{T}_3\vec{f}^{P} = \underline{F}_Q(\underline{F}^{T}\underline{F})^{-1}\underline{F}_P^{T}\vec{f}^{P} = \frac{\|\vec{f}^{P}\|^2}{\|\vec{f}\|^2}\vec{f}^{Q}$$

and our proof is complete.

Appendix E: *A "Good" Interconnect Relaxation Parameter*

Here we show that the relaxation parameter in (13) performs "well" for synchronous operation.

Synchronous Operation

Convergence is maximally accelerated by choosing the value of θ that minimizes the spectral radius of $\underline{T}_4^{\theta}$. Equivalently, we desire to minimize the ℓ_∞ norm of the eigenvalue of sequence $\{\lambda_1^{\theta},\lambda_2^{\theta},\ldots,\lambda_Q^{\theta}\}$. Let λ_{min} and λ_{max} denote the minimum and maximum eigenvalues of $\underline{T}_4$.

The relaxation that minimizes the spectral radius will result in

$$-\lambda_{min}^{\theta} = \lambda_{max}^{\theta}$$

so that, from (12a), we must satisfy

$$-[\theta(\lambda_{min} - 1) + 1] = [\theta(\lambda_{max} - 1) + 1]$$

Thus, we would liks to choose

$$\theta = [1 - \frac{1}{2}(\lambda_{max} + \lambda_{min})]^{-1}$$

Doing so, however, necessitates the computation of λ_{max} and λ_{min}.

An alternate suboptimal choice of θ that can be more easily computed arises from minimizing the ℓ_2 norm of the relaxed eigenvalue sequence. Define

$$f(\theta) = \sum_{r=1}^{Q}[\lambda_r^{\theta}]^2 = \sum_{r=1}^{Q}[\theta(\lambda_r - 1) + 1]^2$$

Minimizing by differentiation gives

$$\sum_{r=1}^{Q}(\lambda_r - 1)[\theta(\lambda_r - 1) + 1] = 0$$

From which we conclude

$$\theta = \frac{\sum_{r=1}^{Q}(1-\lambda_r)}{\sum_{r=1}^{Q}(1-\lambda_r)^2} = \frac{tr(\underline{I} - \underline{T}_4)}{tr[(\underline{I} - \underline{T}_4)^2]}$$

APPENDIX F: *Adding Neurons to the Hidden Layer Improves the Convergence Rate*

Partition the augmented library matrix as

$$\underline{F}_+ = \begin{bmatrix} \underline{F}_P \\ \hline \underline{F}_H \\ \hline \underline{F}_Q \end{bmatrix} \tag{F1}$$

where $\underline{F}_H$ contains the state of the hidden layers. Then

$$\begin{aligned} \underline{F}_+^T \underline{F}_+ &= \underline{F}_P^T \underline{F}_P + \underline{F}_Q^T \underline{F}_Q + \underline{F}_H^T \underline{F}_H \\ &= \underline{B} + \underline{F}_H^T \underline{F}_H \\ &\equiv \underline{A} \end{aligned} \tag{F2}$$

where

$$\underline{B} = \underline{F}_P^T \underline{F}_P + \underline{F}_Q^T \underline{F}_Q$$

Manipulation of (F2) gives

$$\underline{B}^{-1} - \underline{A}^{-1} = \underline{B}^{-1} \underline{F}_H^T \underline{F}_H \underline{A}^{-1} \tag{F3}$$

The augmented interconnect matrix corresponding to (F1) is

$$\underline{T}_+ = \underline{F}_+ (\underline{F}_+^T \underline{F}_+)^{-1} \underline{F}_+^T$$

Let $\underline{T}_4^+$ denote the lower right Q by Q partition of $\underline{T}^+$. Then

$$\underline{T}_4^+ = \underline{F}_Q \underline{A}^{-1} \underline{F}_Q^T$$

The spectral radius of $\underline{T}_4^+$ dictates the convergence rate of the APNN with hidden neurons. Without hidden neurons, convergence is dictated by the spectral radius of

$$\underline{T}_4 = \underline{F}_Q \underline{B}^{-1} \underline{F}_Q^T$$

Thus

$$\begin{aligned} \underline{T}_4 - \underline{T}_4^+ &= \underline{F}_Q (\underline{B}^{-1} - \underline{A}^{-1}) \underline{F}_Q^T \\ &= \underline{F}_Q (\underline{B}^{-1} \underline{F}_H^T \underline{F}_H \underline{A}^{-1}) \underline{F}_Q^T \end{aligned} \tag{F4}$$

where, in the second step, we have used (F3).

The right hand side of (F4) is clearly positive semi-definite. Thus, for any vector $\vec{x}$,

$$\vec{x}^T \underline{T}_4 \vec{x} \geq \vec{x}^T \underline{T}_4^+ \vec{x}$$

Equality holds if the hidden neurons do not increase the rank of $\underline{F}_P$. The spectral radius of $\underline{T}_4^+$ is then clearly smaller than that of $\underline{T}_4$.

The proof can be straightforwardly altered to show that adding more neurons to an established hidden layer further improves convergence.