K.F. Cheung, R.J. Marks II and L.E. Atlas, "Neural net associative memories based on convex set projections", Proceedings of the IEEE First International Conference on Neural Networks, San Diego, June 1987, pp.III-245-III252.

Neural Net Associative Memories Based on Convex Set Projections

Kwan F. Cheung, Seho Oh, Robert J. Marks II, Les E. Atlas
Interactive Systems Design Lab
University of Washington, Seattle, Wa 98195.

I. INTRODUCTION

This paper continues further results on the analysis of a continuous level neural networks proposed by Marks [1]. The net stores continuous level library vectors in the neural interconnects. When a portion of a library vector is imposed on a subset of the neurons, the networks iteratively extrapolates the remainder. Nguyen and Holt [2] have suggested using stochastic processing to implement the network. Optical architectures have also been proposed [3].

A brief review of the extrapolation neural net is given in the next section. We demonstrate that, if there is no restoration ambiguity, the network converges when only one neuron in the network is allowed to change state at a time, or if the net runs synchronously. The performance of the net is also examined for the cases of insufficient and improper excitation. The network is shown to be able to be trained one library vector at a time using a Gram-Schmidt procedure. Lastly, imposition of further nonlinear constraints is suggested as a technique to further increase the convergence rate.

II. THE EXTRAPOLATION NEURAL NETWORK

In this section, we established the notation for the extrapolation neural network. Consider a set of N continuous level linearly independent vectors of length L > N: $\{ \vec{f}_n \mid 0 \le n \le N \}$. We form the library matrix

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

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 demotes transposition. We divide the nodes 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 node at time M. If the kth node falls into the known category, its state is clamped to the known value (i.e. $S_{k,m} = f_k$). The states of 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 simulaneously (i.e. $S_k = S_{k,M-1}$), 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, for either synchronous or asynchronous operation, 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 rank. By strong convergence, 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 III. Both linear and nonlinear alteration techniques to improve the network's convergence rate is in section VI.

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 states. That is $P \geq N$.

Partition Notation

The partition of clamped and floating nodes can change from application to application. The analysis in this paper, however, will be restricted to a single application. Thus, without loss of generality, we will assume that neurons 1 through P are clamped and the remaining nodes are floating. We adopt the vector partitioning notation

$$\vec{i} = [\vec{i}_P | \vec{i}_Q]^T$$

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

$$\underline{F}_P = [\ \vec{f}_{1P} | \vec{f}_{2P} | \ldots | \vec{f}_{NP}]$$

Using this partition notation, we can define nodal clamping operator by

$$\underline{\eta}\, \vec{i} = [\vec{f}_P | \vec{i}_Q]^T$$

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

Partitioning 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 is $\underline{T}_2$ and $\underline{T}_4$. Furthermore $\underline{T}_1 = \underline{T}_3^T$.

III. CONVERGENCE PROOFS

In this section, we prove convergence of the networks for synchronous operation and for a case where only one neuron at a time changes state. Both proofs require $\underline{F}_P$ be full rank. The behaviour of the network when $\underline{F}_P$ is not full rank is also addressed.

A. Synchronous Operation

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

$$\vec{i}_M = \underline{T}\,\vec{s}_M$$

The known 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

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

This operation can best be visualized in an L dimensional Hilbert space. The $\underline{T}$ matrix orthogonally projects any vector onto N dimensional subspace, T, formed by the closure of the library vectors [4]. The clamping operator, η, orthogonally projects onto 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 Von Neumann's alternating projection theorem [5,6], alternating orthogonal projections between two linear varieties strongly converge to a point common to both. Clearly, the library vector $\vec{f}$ is common to both T and η. The requirement that $\underline{F}_P$ is full rank assures that $\vec{f}$ is the only point of intersection 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

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

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 equivalently write (4) as

$$\vec{s}_{M+1,Q} = \left[\begin{array}{c|c} \underline{T}_3 & \underline{T}_4 \end{array}\right] \left[\begin{array}{c} \vec{f}_P \\ \hline \vec{s}_{M,Q} \end{array}\right]$$

or

$$\vec{s}_{M+1,Q} = \underline{T}_3 \vec{f}_P + \underline{T}_4 \vec{s}_{M,Q}$$

If the spectral radius of $\underline{T}_4$ is less than one, the steady state solution of this difference equation can be written as

$$\vec{s}_{\infty,Q} = (\underline{I} - \underline{T}_4)^{-1} \underline{T}_3 \vec{f}_P \tag{5}$$

We have shown that, if $\underline{F}_P$ is full rank,

$$\vec{s}_{\infty,Q} = \vec{f}_Q$$

That is, the steady state solution is the extrapolation of the library vector $\vec{f} = [\vec{f}_P \mid \vec{f}_Q]^T$.

B. Sequential Operation

An asynchronous network can be defined as one in which two or more neurons do not change state at the same time [7]. Subsumed in this concept is sequential operation wherein neural states are updated periodically in indexed order. That is, neuron 1 is allowed to change. Then neuron 2 is updated. After every (floating) neuron is allowed to change, the procedure is iteratively repeated.

The convergence proof for sequential operation is based on (5) which can be written as

$$\underline{A}\vec{y} = \vec{b}$$

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

$$x_{M+1,i} = - \sum_{k=1}^{i-1} a_{ik} x_{M+1,k} + (1 - a_{ii}) x_{M,i} - \sum_{k=i+1}^{L} a_{ik} x_{M,k} + g_i$$

converges to the desired vector, $\vec{y}$, if the spectral radius of $\underline{T}_4$ is less than one. The proof is similar to that for the Gauss-Seidel algorithm [8] but is not included here due to space limitations.

C. Results of Noncompliance with Conversion Criteria

1. The Underdetermined Case:

If $\underline{F}_P$ is not full 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, $\vec{s}_o$ [9]. Equivalently, $\vec{s}_\infty$ is the orthogonal projection of $\vec{s}_o$ onto $\mathcal{V}$. This result is geometrically illustrated in Fig. 1.

2. 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 Fig. 3, the networks will converge to that point on the linear variety closest to the subspace [10].

When T and η do intersect, the sum of the inputs for the clamped nodes approaches the clamped values. This is not the case for non- intersection. (In fig. 2, 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 thus implies improper clamped values.

IV. **LEARNING**

The equation for the interconnect matrix in (1) is computationally unacceptable. A better procedure is to train the neural network one library vector at a time. The result is a procedure for teaching the neural networks new library vectors.

Assume we have an interconnect matrix, $\underline{T}$, and wish to update the interconnects corresponding to a new library vector, $\vec{f}$. As illustrated in Fig. 3, $\underline{T}\vec{f}$ projects $\vec{f}$ onto T and

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

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

We wish to extend the dimension of 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

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

now projects any L tuplet onto the new subspace formed by the closure of T and $\vec{\varepsilon}$ or, equivalently, T and $\vec{f}$. This procedure 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 direction dictated by computational or other noise. Thus, in order to assure the networks is learning something useful, it is thus advisable to compare $\vec{\varepsilon}^T\vec{\varepsilon}$ to some appropriate threshold prior to updating [11].

V. IMPROVING CONVERGENCE

A classic technique to improve iterative convergence algorithms is relaxation. Such techniques are discussed elsewhere [1,12].

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

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

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.

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 learning is completed, we have

$$\alpha_k = \min_{1 \le n \le N} f_{nk}$$

and

$$\beta_k = \max_{1 \le n \le N} f_{nk}$$

Upper and lower thresholding the elements of a vector at preset values can be viewed as the projection of the vector onto a box the dimensions of which are specified by the threshold values. As illustrated in Fig. 4, the convergence rate of the net can be improved by this procedure.

Convergence can be proven by an appeal to the results of Youla and Webb [10] who show that alternately projecting between two or more intersecting convex sets[1] results in convergence to a point common to all of the sets. Since the box, linear variety and subspace are all convex, the theorem is applicable here. Furthermore, since we've required a single point of intersection between T and η, the three set projection procedure converges to a single point.

---

[1] A set $\mathcal{C}$ is convex if $\alpha x + (1-\alpha) y \in \mathcal{C} \;\; \forall \, x,y \in \mathcal{C}$ over the interval $0 \le \alpha \le 1$.

VI. CONCLUDING REMARKS

We have examined various properties of a continuous level neural network capable of extrapolating stored library vectors. Topics on which we will be reporting in the future include an analysis of the effects of relaxation on convergence, an analogous table lookup neural network, alternate but algorithmically equivalent computational architectures, fault tolerance properties and effects of input, system and detector noise.

ACKNOWLEDGEMENTS

This work was supported by the Boeing High Technology Center as administered through the Washington High Technology Center and also by the SDIO/IST's program in Ultra High Speed Computing administered through ONR in conjunction with Texas Tech University. Les. E. Atlas was also supported by an NSF PYI award. The authors gratefully acknowledge the inputs of Dziem Nguyen, Fred Holt and Don Wunsch of the Boeing High Technology center and Jim Ritcey of the Interactive Systems Design Lab.

REFERENCES

1. R.J.Marks II, Appl. Opt. **26** p.2005 (1987).

2. D.Nguyen, F.Holt, Proc. IEEE First Annual International Conference on Neural Networks, 21-24 June,1987; San Diego.

3. R.J.Marks II, L.E.Atlas, K.F.Cheung, Proc. 14th Congress of the ICO, 24-27 Aug.,1987, Quebec, Canada.

4. G.Strang, Linear Algebra and Its Applications, (Academic Press, New York, 1980).

5. W.Duane Montgomery, Opt. Lett. p.1 (1982).

6. J.Von Neumann, The Geometry of Orthogonal Spaces,(Princeton U. Press, Princeton, N.J. 1950).

7. K.F. Cheung, R.J. Marks II, L.E. Atlas, "Synchronous vs. Asynchronous Behaviour of Hopfield's CAM Neural Net", (submitted for publication.)

8. A.Ralston, P.Rabinowitz, A First Course in Numerical Analysis, (2nd Edition, McGraw Hill, 1965).

9. M.R.Civanlar, H.J.Trussel, IEEE Trans. ASSP, **ASSP-34**., p.919 (1986).

10. M.Goldberg, R.J.Marks II, IEEE Trans. CAS, **CAS-32**, p.647 (1985).

11. G.Eichman, M.Stojancic, Appl. Opt., **26**, p.1911 (1987).

12. D.C.Youla, H.Webb, IEEE Trans. Med. Imaging, p.81 (1982).

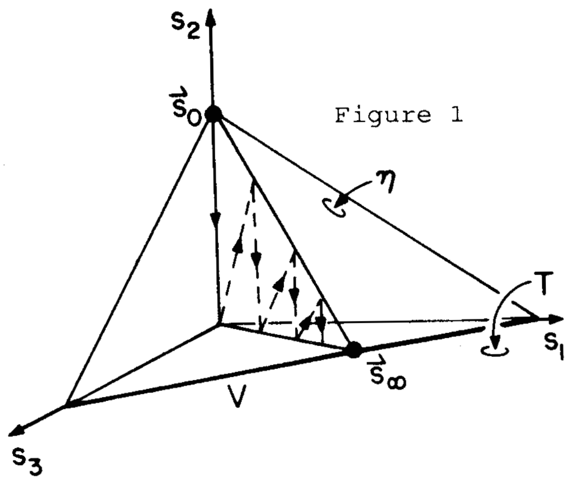

Figure 1

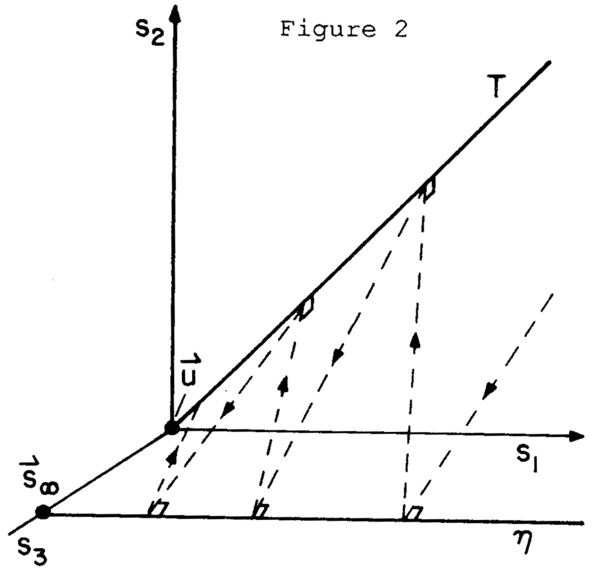

Figure 2

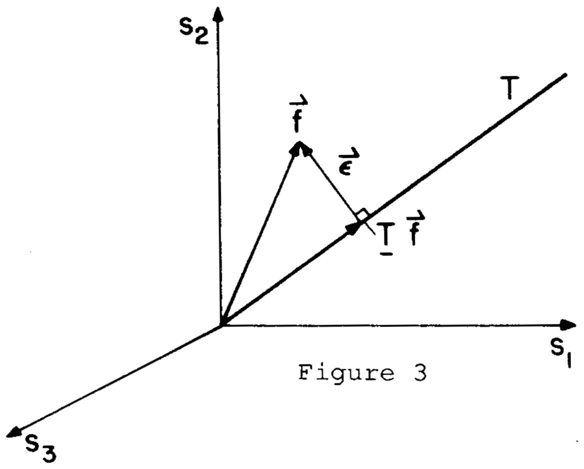

Figure 3

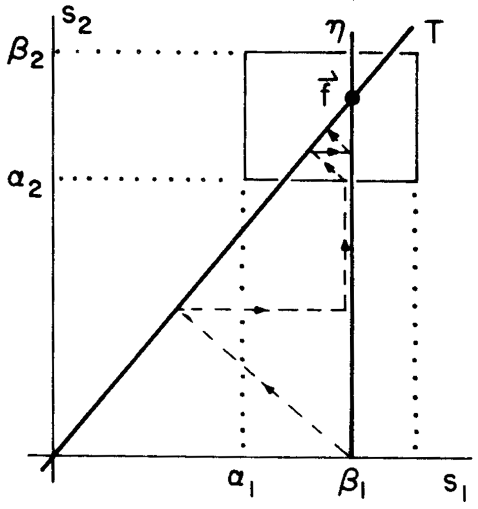

Figure 4