A NEURAL NETWORK FOR COMPUTING THE PSEUDO-INVERSE OF A MATRIX AND APPLICATIONS TO KALMAN FILTERING

Mathew C. Yeates

California Institute of Technology
Jet Propulsion Laboratory
Pasadena, California 91109

ABSTRACT

A single layer linear neural network for associative memory is described. The matrix which best maps a set of input keys to desired output targets is computed recursively by the network using a parallel implementation of Grevilles algorithm. This model differs from the Perceptron in that the input layer is fully interconnected leading to a parallel approximation to Newtons algorithm. This is in contrast to the steepest descent algorithm implemented by the Perceptron. By further extending the model to allow synapse updates to interact locally, a biologically plausible addition, the network implements Kalman filtering for a single output system.

A NEURAL NETWORK FOR COMPUTING THE PSEUDO-INVERSE OF A MATRIX AND APPLICATIONS TO KALMAN FILTERING

I. INTRODUCTION

One result of the current interest in neural modeling, [2][5][6][9][11], has been the emergence of structures that, while not strict models of biological neural systems, are, nevertheless, interesting in their own right for their ability to do complex processing in a massively parallel way. This correspondence describes such a structure, a single layer linear feedforward network, as well as its ability to implement several classical signal processing algorithms.

Suppose, given the vector pairs $(\bar{x}_i, \bar{y}_i)$ i=1..p with $\bar{x}_i \in R^n$ $\bar{y}_i \in R^m$, it is desired to find an m x n matrix W such that $Wx_i = y_i$ for all i. If such a matrix exists, the linear system described by $\bar{y} = W\bar{x}$ acts as an associative memory where the $\bar{y}_i$ can be thought of as stored memories and the $\bar{x}_i$ are the stimuli that retrieve them.

One neural approach is to form W as the sum of outer products of the input vectors with the output vectors, the Correlation Matrix Memory [8]. This scheme is can be easily implemented by a neural network with W being the matrix of connectivity between the inputs and outputs but perfect recall is realized only when the input vectors are orthogonal. Attempts at using this type of memory are described in [8][3].

A better choice for W involves calculating the pseudo-inverse of the matrix X with columns constructed with the input vectors. The mapping problem can formulated using matrix terminology by letting $X = [\bar{x}_1, \bar{x}_2, .., \bar{x}_p]$ and $Y = [\bar{y}_1, \bar{y}_2, .., \bar{y}_p]$. The

input output pairs form the columns of X and Y and W is sought that minimizes $||Y-WX||_E$, where $||\ ||_E$ is the Euclidean matrix norm satisfying $||A||_E = \sqrt{tr\ (AA^T)}$. In general, X may not be invertible and the best solution [10] is given by $W = YX^+ + Z - ZXX^+$ where X^+ is the generalized notion of an inverse, called the Penrose Pseudoinverse, and Z is an arbitrary matrix of the same dimension as X. Taking Z=0 for uniqueness, the solution of minimum norm, $W = YX^+$, is obtained.

Several researchers have investigated the use of the pseudoinverse to to implement the desired mapping. See [12]. However, these attempts involve an off-line calculation and only after the the desired W is found is it stored in a neural network. The contribution of this correspondence is to show that the neural network itself can quickly learn the desired mapping.

Calculating X^+ involves a matrix inversion when either the rows or columns of X are independent. Specifically,

$$X^+ = X^T(XX^T)^{-1} \qquad rows \quad independent$$

$$X^+ = (X^TX)^{-1}X^T \qquad columns \quad independent$$

While any matrix inversion routine could be used to solve for X^+ in either of these cases, a recursive algorithm is desired for implementation on a neural network. This is provided for by a special case of Greville's algorithm [8]. W can be generated by

$$W_k = W_{k-1} + \frac{(\bar{y}_k - W_{k-1}\bar{x}_k)\bar{x}_k^T\Psi_{k-1}^T}{1 + \bar{x}_k^T\Psi_{k-1}\bar{x}_k} \tag{1.1}$$

$$\Psi_k = \Psi_{k-1} - \frac{\Psi_{k-1}\bar{x}_k\bar{x}_k^T\Psi_{k-1}^T}{1 + \bar{x}_k^T\Psi_{k-1}\bar{x}_k} \quad k = 0 \ \cdots \ p$$

whenever x_k is linearly dependent on the previous input vectors and the rows of X_{k-1} are independent. Because these assumptions are not true in general, the original problem is modified slightly. Instead of asking for W that maps $\bar{x_i}$ to $\bar{y_i}$ for i=1..p, it is also asked that W map the vectors εe_i i=1..n to the zero vector $\bar{0}$. Here, ε is a parameter and e_i is the i'th standard basis function. Intuitively, if ε is small, the perturbation is expected to have a small overall effect on the original problem. With these changes, the problem becomes one of finding a W that minimizes $||Y - WX||_E$ with $X = [\varepsilon I_{nxn} : \bar{x}_{n+1}, \bar{x}_{n+2}, .., \bar{x}_{n+p}]$ and $Y = [0_{mxn} : \bar{y}_{n+1}, \bar{y}_{n+2}, .., \bar{y}_{n+p}]$. Now the conditions are met and (1.1) can be used for $k = n \cdots n+p$ with initial conditions $\Psi_n = (X_n X_n^T)^{-1} = \frac{1}{\varepsilon^2} I$ and $W_n = 0$. This modified algorithm is suitable for implementation on a neural structure, as is shown next.

II. A NEURAL MODEL

While the algorithm of section I is well known, the contribution of this paper is its implementation on the parallel architecture depicted in Figure 1. Note that the input layer is fully connected and there is a node, the scaling node, connected to every other node.

As before, W stands for the connection strengths between the input and output layer. Next, let $\Psi = \{\psi_{i,j}\}_{i=1..n, j=1..n}$ stand for the connection strengths in the input layer with $\psi_{i,j}$ being the weight on the line connecting the input nodes i and j. The net is started with $W = 0$ and $\Psi = \frac{1}{\varepsilon^2} I_{nxn}$ so that originally, each input node only affects itself. In response to the presentation of the pair $(\bar{x}_k, \bar{y}_k)$, the following steps are

performed:

1) $\bar{x}_k$ is given to the input nodes.

2) A. $\bar{x}_k$ is fed through W_{k-1} and the difference $\bar{y}_k - W_{k-1}\bar{x}_k$ is held by the output nodes.

   B. At the same time, $\bar{x}_k$ is fed through the interconnected input layer so that now the input layer has as its value $\Psi_{k-1}\bar{x}_k$. As a new value is entering an input node, this incoming value is multiplied by the present value and the result is sent to the scaling node which performs a summation. After this step, the scaling node has value $\bar{x}_k^T\Psi_{k-1}\bar{x}_k$.

3) The scaling node adds 1 to its present value, takes the square root and broadcasts the result to every node simultaneously. The value of each node is multiplied by the incoming scaling node value. The effect of this is that now $\frac{1}{\sqrt{1+\bar{x}_k^T\Psi_{k-1}\bar{x}_k}}\Psi_{k-1}\bar{x}_k$ is held in the input nodes and $\frac{1}{\sqrt{1+\bar{x}_k^T\Psi_{k-1}\bar{x}_k}}(\bar{y}_k - W_{k-1}\bar{x}_k)$ is held in the output nodes.

4) A. Hebbian learning occurs for the weights connecting input to output i.e.

$$W_k = W_{k-1} + \frac{(\bar{y}_k - W_{k-1}\bar{x}_k)}{\sqrt{1 + \bar{x}_k^T\Psi_{k-1}\bar{x}_k}} \frac{(\Psi_{k-1}\bar{x}_k)^T}{\sqrt{1 + \bar{x}_k^T\Psi_{k-1}\bar{x}_k}} = W_{k-1} + \frac{(\bar{y}_k - W_{k-1}\bar{x}_k)\bar{x}_k^T\Psi_{k-1}^T}{1 + \bar{x}_k^T\Psi_{k-1}\bar{x}_k}$$

   B. "Anti-Hebbian" learning occurs at the input layer, i.e.

$$\Psi_k = \Psi_{k-1} - \frac{(\Psi_{k-1}\bar{x}_k)}{\sqrt{1 + \bar{x}_k^T\Psi_{k-1}\bar{x}_k}} \frac{(\Psi_{k-1}\bar{x}_k)^T}{\sqrt{1 + \bar{x}_k^T\Psi_{k-1}\bar{x}_k}} = \Psi_{k-1} - \frac{\Psi_{k-1}\bar{x}_k\bar{x}_k^T\Psi_{k-1}^T}{1 + \bar{x}_k^T\Psi_{k-1}\bar{x}_k}$$

After these four steps, the net has performed the algorithm (1.1) in parallel, in a manner consistent with common neural learning schemes. It should be noted that the net requires only one presentation of each vector pair thus distinguishing it from other iterative schemes. Some extensions to the model are described next.

III. VARIATIONS OF THE BASIC MODEL

In this is section it is shown that by slightly modifying the neural network, the Interconnected Adaptive Linear Combiner (ICALC), a sequential regression algorithm and Kalman filtering can be implemented.

1) Suppose n different signals are sampled at a time $t = k$ and the values taken are denoted by $\bar{x}_k = [x_{1k}, x_{2k}, \cdots, x_{nk}]^T$. Further, suppose this vector is input to a linear system whose output is a weighted sum of the inputs. Then the output of the system at time k is given by $y_k = \bar{w}^T \bar{x}_k$ with $\bar{w} = [w_1, w_2, \cdots, w_n]^T$. If t_k is a target sequence and $e_k = t_k - y_k$, then the instantaneous squared error is $e_k^2 = t_k^2 + \bar{w}^T \bar{x}_k \bar{x}_k^T \bar{w} - 2t_k \bar{x}_k^T \bar{w}$. The $\bar{w}$ which minimizes ε can be found iteratively by

$$\bar{w}_{k+1} = \bar{w}_k + 2\eta(t_k - \bar{w}_k^T \bar{x}_k)\bar{x}_k \tag{3.1}$$

a gradient descent procedure implemented by Widrow's Adaptive Linear Combiner [14]. While convergence is guaranteed for small enough η [14], difficulties in the use of this algorithm are well known. Better performance is achieved through the use of Newtons algorithm which can be approximated by,

$$\begin{aligned} \bar{w}_{k+1} &= \bar{w}_k + \eta Q_k^{-1} \bar{x}_k (t_k - \bar{w}_k^T \bar{x}_k) \\ Q_k^{-1} &= \frac{1}{\alpha} Q_{k-1}^{-1} - \frac{(Q_{k-1}^{-1}\bar{x}_k)(Q_{k-1}^{-1}\bar{x}_k)^T}{\alpha(\alpha + \bar{x}_k^T Q_{k-1}^{-1} \bar{x}_k)} \end{aligned} \tag{3.2}$$

It is easy to see how these update equations can be implemented on a slightly modified

ICALC by having the scaling node compute $\sqrt{\alpha(\alpha+\bar{x}_k^T Q_{k-1}^{-1}\bar{x}_k)}$, allowing the input-output weights to adapt at a slower rate η, and introducing a "forgetting factor" $\frac{1}{\alpha}$.

The algorithm described by (3.2) is called the Sequential Regression Algorithm (SER) [1][4] and its derivation is more fully described in [13]. The latter reference includes experimental justification for using SER over (3.1) as well as giving some bounds on the size of the learning parameter η needed for convergence. It also describes several areas where the Adaptive Linear Combiner has been used e.g. system identification, modeling, and adaptive control. Another application, mentioned in [7], is the use of the Adaptive Linear Combiner for the coding of speech signals.

2).Consider a time invariant, single output linear system written in state space form as

$$\bar{w}_{k+1} = \bar{w}_k \tilde{A} \quad \tilde{A} = A^T \tag{3.3}$$

$$y = \bar{w}_k \bar{x}_k + n_k$$

$\bar{w}$ represents the internal state of the system, y is the output, and the matrix $\tilde{A}$ is the n x n state transition matrix. The sequence n_k is white noise with $E[n_{k+r} n_k] = N\delta(r) \quad N>0$. Now suppose an observer, after measuring y, wishes to estimate the state of the system. The linear minimum variance estimate is given by the Kalman Filter

$$\bar{w}_f = \bar{w}_{k-1} + \frac{(y - \bar{w}_k \bar{x}_k)\bar{x}_k^T \Psi_{k-1}^T}{N + \bar{x}_k^T \Psi_{k-1} \bar{x}_k} \tag{3.4}$$

$$\Psi_f = \Psi_{k-1} - \frac{\Psi_{k-1}\bar{x}_k \bar{x}_k^T \Psi_{k-1}^T}{N + \bar{x}_k^T \Psi_{k-1} \bar{x}_k}$$

$$\bar{w}_k = \bar{w}_f \tilde{A}$$

$$\Psi_k = \tilde{A}^T \Psi_f \tilde{A}$$

Note that $\bar{w}$ is a row vector. Now, the first two equations of (3.4) are (1.1) for the single output case. For ICALC to fully implement Kalman filtering the weights must further change according to the last two equations.

For general A, it can be assumed that A is block diagonal. Any matrix can be reduced to this form by a similarity transformation. Because of the structure of A, the weights are partitioned into separate sets, the elements of each set interacting only with other elements in that set. Further, each block has a structure allowing only a small number of interactions. For instance, suppose $\tilde{A}$ has the form

$$\begin{bmatrix} \tilde{A}_1 & O & O \\ O & \tilde{A}_2 & O \\ O & O & \tilde{A}_3 \end{bmatrix}$$

with $\tilde{A}_1$, $\tilde{A}_2$, $\tilde{A}_3$ square matrices of size n_1, n_2, n_3, $n_1 + n_2 + n_3 = n$. Letting $\bar{w}_k^{(1)}$ be the vector formed with the first n_1 components of $\bar{w}_k$ and doing the same for $\bar{w}_k^{(2)}$ and $\bar{w}_k^{(3)}$, $\bar{w}_k$ can be written as $\bar{w}_k = [\bar{w}_k^{(1)} : \bar{w}_k^{(2)} : \bar{w}_k^{(3)}]$. With the same partitioning of $\bar{w}_f$ the third equation of (3.4) takes the form $\bar{w}_k^{(i)} = \bar{w}_f^{(i)} \tilde{A}_i \quad i=1...3$. In a similar manner, Ψ_k is partitioned in blocks $\Psi_k^{(ij)}$ with n_i rows and n_j columns and the last equation becomes $\Psi_k^{(ij)} = \tilde{A}_i^T \Psi_f^{(ij)} \tilde{A}_j \quad i,j=1...3$.

The similarity transformation which reduces A to block diagonal form can also be chosen so that the blocks are nearly diagonal. Each $\tilde{A}_i$ has the form

$$\begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & * & \\ & & * & 1 \\ & & & \lambda_i \end{bmatrix}$$

Considering $\bar{w}_k^{(1)}$ and writing $\bar{w}_k^{(1)} = [w_1^{(1)}, w_2^{(1)}, \ldots, w_{n_1}^{(1)}]$, and $\bar{w}_f^{(1)} = [w_{f1}^{(1)}, w_{f2}^{(1)}, \ldots, w_{fn_1}^{(1)}]$ the weights for the first set are updated according to

$$w_i^{(1)} = \lambda_1 w_{fi}^{(1)} + w_{f(i+1)}^{(1)} \quad i=2...n_1 \tag{3.5}$$

$$w_1^{(1)} = \lambda_1 w_{f1}^{(1)}$$

The extension to the sets $\bar{w}_k^{(2)}$ and $\bar{w}_k^{(3)}$ is straightforward. To illustrate the updating for Ψ, consider the n_1 by n_2 block $\Psi^{(12)}$. Let $\Psi_k^{(12)} = \{\psi_{i,j}^{(12)}\}_{i=1..n_1,\ j=1..n_2}$ and $\Psi_f^{(12)} = \{\psi_{f,i,j}^{(12)}\}_{i=1..n_1,\ j=1..n_2}$. Now, using $\Psi_k^{(12)} = \tilde{A}_1^T \Psi_f^{(12)} \tilde{A}_2$, the update equations for this block can be written

$$\psi_{i,j}^{(12)} = \begin{cases} \lambda_2(\lambda_1\psi_{f,i,j}^{(12)} + \psi_{f,i-1,j}^{(12)}) + \lambda_1\psi_{f,i,j-1}^{(12)} + \psi_{f,i-1,j-1}^{(12)} & i=2...n_1,\ j=2...n_2 \\ \lambda_2(\lambda_1\psi_{f,i,1}^{(12)} + \psi_{f,i-1,j}^{(12)}) & i=2...n_1,\ j=1 \\ \lambda_2(\lambda_1\psi_{f,1,1}^{(12)}) & i=1,\ j=1 \\ \lambda_2(\lambda_1\psi_{f,1,j}^{(12)}) + \lambda_1\psi_{f,1,j-1}^{(12)} & i=1,\ j=2...n_2 \end{cases} \tag{3.6}$$

If the nodes are arranged so that the weights have the same spatial orientation as in the Ψ matrix, then the change in a weight depends only on three neighboring weights. For instance, if a column of Ψ is thought of as a node, with connections from other nodes occurring at the column entries, the interactions are local. This interdependence of weight changes, although missing from most neural models, is biologically plausible; one does not expect the changes in one neuron to not affect other neurons in close proximity. Adding this simple feature makes Kalman filtering possible.

The research reported here was sponsored by the Strategic Defense Initiative Organization/Innovative Science and Technology through an agreement with the National Aeronautics and Space Administration (NASA).

REFERENCES

[1] N. Ahmed, D. L. Soldan, D. R. Hummels, and D. D. Parikh, "Sequential Regression Considerations of Adaptive Filtering," *Electron. Lett.,* p.446, July 21, 1977.

[2] J. A. Anderson, "Cognitive and Psychological Computation with Neural Models," *IEEE Trans. Syst. Man Cybern.* SMC-13, 799-814, 1983.

[3] L. N. Cooper, "A Possible Organization of Animal Learning and Memory," *Proc. of the Nobel Symposium on Collective Properties of Physical Systems*, Vol. 24, pp 252-264, Academic Press, New York, 1974.

[4] D. Graupe, *Identification of Systems,* Van Nostrand Reinhold, New York (1972).

[5] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley, "Boltzman Machines: Constraint Satisfaction Machines That Learn," Technical Report CMU-CS-84-119, Carnagie- Mellon University, May 1984.

[6] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. Natl. Acad. Sci.* USA, Vol. 79, 2554-2558, April 1982.

[7] N. S. Jayant and P. Noll, *Digital Coding of Waveforms,* Prentice-Hall, N.J. (1984).

[8] T. Kohonen, *Self-Organization and Associative Memory,* Springer-Verlag, Berlin (1984).

[9] R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine,* April 1987.

[10] R. Penrose, *Proc. Cambridge Philos. Soc.* 51, 406 (1955).

[11] D. E. Rummelhart and J. L. McClelland (Eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol 1: Foundations.* MIT Press (1986).

[12] B. Telfer and D. Casasent, "Updating Optical Pseudoinverse Associative Memories," *Applied Optics*, Vol. 28, 2518-2528, July 1989.

[13] B. Widrow and S. Stearns, *Adaptive Signal Processing*, Prentice-Hall, N.J. (1985).

[14] B. Widrow, "Adaptive Filters," in *Aspects of Network Switching Theory*, R. E. Kalman and N. De Claris (Eds.). Holt, Rinehart and Winston, New York (1970).