OBTAINING INITIAL PARAMETER ESTIMATES FOR NONLINEAR SYSTEMS: COMPARING ASSOCIATIVE MEMORY AND NEURAL NETWORK APPROACHES

JAMES E. MOORE II 1
MOON KIM
JONG-GOOK SEO
School of Urban and Regional Planning,
and
ROBERT KALABA 1
Departments of Biomedical Engineering, Electrical Engineering, and Economics,
University of Southern California, Los Angeles, CA 90089

Abstract

Many optimisation procedures presume the availability of an initial approximation in the neighborhood of a local or global optimum. Unfortunately, finding a set of good starting conditions is a nontrivial objective. Previous papers [7, 8, 12, 13, 20, 21] describe procedures that use various multicriteria and recurrent associative memories to identify initial approximations of solutions to parameter identification and operations research problems. In this paper, we compare the performance of Kalaba and Tesfatsion's [12] multicriteria associative memory to those of a recurrent associative memory [8, 21] and a feed-forward neural network trained with the same data.

1. INTRODUCTION

Recent work by Kalaba and Tesfatsion [12] and Kalaba and Udwadia [13] investigate the use of multicriteria associative memories for parameter identification in economic and mechanical systems. These objectives are conventionally treated as nonlinear optimization problems; and are usually solved numerically by iterative procedures, most often by variants of the Gauss-Newton method [2, 28]. These and competing procedures such as simplicial search methods [1, 11] require a good initial estimate of the parameter vector. If the initial estimate is not within ten to twenty percent of the true value, even highly convergent procedures are likely to diverge, or to converge to a spurious minimum. Unfortunately, the question of how to generate parameter estimates of sufficient quality to take advantage of the standard numerical procedures for solving nonlinear least squares and maximum likelihood problems has received limited attention in the various branches of the systems literature.

2. VECTOR MAPPING

Parameter identification is most generically characterized as a class of vector mapping problems. Given a vector describing the trajectory of system states, what is the underlying parameter vector? Associative memories and neural networks both address this pair-association problem at a general level. Does there exist an associative memory M [14, 22] or a neural network N [5, 26, 27] that will map a finite set of arbitrarily selected stimulus vectors to the corresponding set of response vectors?

Treating a system time path and underlying set of parameter values as a stimulus-response pair provides a very useful perspective. For each of K training cases, let the

1 authors to whom correspondence may be addressed.
stimulus vector $s_k$ of dimension $p \times 1$, and the response vector $r_k$ of dimension $q \times 1$, be specified.

### 2.1 Linear Associative Memories

In the case of a simple associative memory, the objective is to determine an associative memory matrix $M^*$ of dimension $q \times p$ such that $M^* \cdot s_k$ will equal $r_k$, as nearly as possible for $k = 1, 2, \ldots, K$. See Figure 1. The hope is that, given a test stimulus vector $s_{k+1}$ not included in the training set, the product

$$M^* \cdot s_{k+1} = r_{k+1}^*$$  \hspace{1cm} (1)

will provide a good estimate of the true test response vector $r_{k+1}$. If $r_{k+1}^*$ is sufficiently close to $r_{k+1}$, then the initial test estimate can be improved by applying a convergent algorithm.

![Diagram of an Ideal Associative Memory](image)

**Fig. 1**  An Ideal Associative Memory.

Following Kohonen [14], we use the $K$ training cases to form the stimulus matrix $S$, whose $k^{th}$ column is $s_k$, and the response matrix $R$, whose $k^{th}$ column is $r_k$. The matrix $M^*$ is determined by minimizing the $L_2$ norm of the difference matrix $R - M \cdot S$,

$$M^* = \arg\min_M \|R - M \cdot S\|^2$$  \hspace{1cm} (2)

clearly, minimizing the $L_2$ norm minimizes mean square error. Noting that

$$\|M\| = \left[\text{Trace} \left( M^T \cdot M \right) \right]^{1/2}$$  \hspace{1cm} (3)

it follows that

$$\|R - M \cdot S\|^2 = \text{Trace} \left[ (R - M \cdot S)^T \cdot (R - M \cdot S) \right]$$  \hspace{1cm} (4)

$$= \text{Trace} \left( R^T \cdot R + R^T \cdot M \cdot S + S^T \cdot M^T \cdot R + S^T \cdot M^T \cdot M \cdot S \right)$$

Minimizing equation (4) with respect to $M$,

$$\partial \|R - M \cdot S\|^2 / \partial M = \text{Trace} \left( -R \cdot S^T - R \cdot S^T + 2M^* \cdot S \cdot S^T \right)$$

$$= 0$$  \hspace{1cm} (5)
Assuming \( S \cdot S^T \) to be non singular, it follows that

\[
M^* = R \cdot S^T \cdot \left[ S \cdot S^T \right]^{-1}
\]  

(6)

More generally, the solution to this problem is

\[
M^* = R \cdot S^+
\]  

(7)

where \( S^+ \), dimension \( K \times p \), is the Moore-Penrose generalized inverse of the rectangular matrix \( S \). Codes for calculating this generalized inverse are available in standard software packages such as SPEAKEASY, MATLAB, and SAS.

Simple linear and nonlinear associative memories have been shown to provide useful estimates in a variety of contexts, including signal processing [9] and linear programming [7]. The generalized inverse \( S^+ \) can be calculated even if \( S \) is not of full rank, but this will generally cause \( S^+ \) to be ill-conditioned. Some elements of \( S^+ \) may be very large relative to the elements in the stimulus and response vectors. This makes any response estimate \( r^* \) very sensitive to noise, including machine rounding error.

2.2 Multicriteria Associative Memories

The problem of ill conditioning can be addressed in a number of ways. Any modification of \( S \) that reduces linear dependency between stimulus vectors will reduce the size of the elements in \( M^* \). This might be accomplished by extending the stimulus vectors with nonlinear combinations of the original elements [7, 23], or by perturbing the stimulus vectors with noise [9, 12]. A more systematic approach is to attach a penalty to the \( L_2 \) norm of the associative memory matrix [12, 13]. Computing such a multicriteria associative memory (MAM) matrix \( M^\alpha \) involves a trade-off between the quality of the approximation provided by applying \( M^\alpha \) to the training data, and the size of the elements in \( M^\alpha \). Defining \( \alpha \) to be a coefficient describing the relative importance of fitting the training data, expression (2) becomes

\[
M^\alpha = \text{arg}\min_M \alpha \cdot \|R - M \cdot S\|^2 + (1 - \alpha) \cdot \|M\|^2
\]  

(8)

and expression (4) becomes

\[
\alpha \cdot \|R - M \cdot S\|^2 + (1 - \alpha) \cdot \|M\|^2 = \text{Trace}[\alpha \cdot (R^T \cdot R + R^T \cdot M \cdot S + S^T \cdot M \cdot R^T + S^T \cdot M^T \cdot M \cdot S) + (1 - \alpha) \cdot M^T \cdot M]
\]  

(9)

Minimizing equation (9) over \( M \),

\[
\frac{\partial \|R - M \cdot S\|^2 + (1 - \alpha) \cdot \|M\|^2}{\partial M} = \text{Trace}[-\alpha \cdot R \cdot S^T - \alpha \cdot R \cdot S^T + 2 \cdot \alpha \cdot M^\alpha \cdot S \cdot S^T + 2 \cdot (1 - \alpha) \cdot M^\alpha]
\]

\[
= 0
\]  

(10)

and thus

\[
M^\alpha = \alpha \cdot R \cdot S^T \cdot \left[ \alpha \cdot S \cdot S^T + (1 - \alpha) \cdot I \right]^{-1}
\]  

(11)

In the special case of

\[
\alpha = 1.0
\]  

(12)

equation (11) reduces to equation (6).
In general, the optimal value of $\alpha$ is unknown. However, Kalaba and Tesfatsion [12] reported that their estimates demonstrate low sensitivity to values on the interval

$$\alpha = [0.1, 0.9]$$

(13)

Alternatively, it is possible to compute generalized inverses via series expansion [10]. If the series is truncated, the magnitude of the elements reported for the generalized inverse can be effectively constrained.

2.3 Recurrent Associative Memories

A recurrent extension of the associative memory approach has been shown to provide much improved estimates of the solutions to constrained optimization problems [8, 20]. In the case of a simple or multicriteria associative memory, applying $M^*$ or $M^A$ to a stimulus vector $s$ produces $r^*$ or $r^A$, estimates of the corresponding response vector. A recurrent associative memory matrix $M^{**}$ is computed by extending the original twining stimulus vectors $s_k$ with $f(r_k^*)$, a nonlinear transformation of the simple associative memory estimate of the corresponding training response vector $r_k$. Given $M^*$, a linear associative memory matrix of dimension $q \times p$, and $R^*$, an estimated training response matrix of dimension $q \times K$, redefine the training stimulus matrix to be

$$S := \left[ \begin{array}{c} S \\ f(R^*) \end{array} \right]$$

(14)

dimension $(p + q) \times K$. The recurrent associative memory matrix $M^{**}$ is computed from equation (7) using the updated definition of $S$. Recurrent training and testing procedures are summarized by the flowchart in Figure 2.

Computational experience here and with other classes of problems reveals that it is important $f(\bullet)$ be a nonlinear function. We obtained very good results in other contexts by relying on the logit and hyperbolic tangent functions frequently used as activation functions in neural networks [8]. In this case, we define $f(\bullet)$ to be the hyperbolic tangent function.

2.4 Feed-Forward Neural Networks

Though more sophisticated than an associative memory, a feed-forward neural network is still a very simple neural structure. In a neural network, hidden layers of artificial neurons use thresholding operations to simulate the learning and distributed processing behavior of biological neurons. Such artificial neural networks are trained by iteratively adjusting coefficients defining the connective strengths between neurons in different layers of the network until a representation of the relationships connecting stimulus and response vectors have been encoded.

The archetype for a simple, nonrecurrent neural network with one hidden layer appears in Figure 3. The S-shaped curves imposed on the network's processing neurons represent a nonlinear transformation of the inputs into outputs by each neuron's activation function. For example, if $x_{i,j,k}$ is the signal passed from unit $i$ in layer $j$ to unit $k$ in layer $j+1$, then the activation function relates the output from unit $k$ to the weighted sum of the inputs to unit $k$,

$$x_{i,j+1,k} = f\left( \sum_i w_{i,j,k} \ast x_{i,j,k} \right)$$

(15)

Activation functions $f(\bullet)$ are typically logistic, tangent hyperbolic, or Cauchy functions. The nonlinearity of cascaded activation functions provides multilayer artificial neural networks with the capacity to encode, via an appropriate combination of weights, functions that are linearly inseparable.
Fig. 2  Computing and Applying a Recurrent Associative Memory (RAM) Matrix $M^{**}$.  

---

Parameter Estimates: Associative Memory and Neural Network Approaches
Minsky and Papert [19] demonstrated that perceptrons (single layer, feed-forward neural networks) could not simulate linearly inseparable functions such as the exclusive-or. Subsequently, Kolmogorov’s [15, 16] proof that "every continuous function of an arbitrarily large number of variables can be represented in the form of a finite superposition of continuous functions of not more than three variables" has been employed as a proof that three-layer feed forward neural networks can approximate (in terms of least squares) any continuous mapping based only on an example of the mapping. However, it is not possible to know in advance just how many hidden units and layers will suffice to capture the relationships of interest. Moreover, obtaining the connective strengths for a feed-forward network is a nonlinear optimization problem for which convergence is not always assured [30]. Back propagation algorithms [26] are necessarily convergent only if the training adjustments made in connective strengths are infinitely small. This assumption is not operationally feasible, and training paralysis is a documented possibility. Simulated annealing [30] can be used to escape the local optima associated with paralysis; but like most general techniques, this is a computationally expensive approach.

A more complete discussion of supervised learning, activation functions, and algorithms used to determine connective strengths between neurons can be found in a number of different sources. The monograph by Rumelhart, Hinton, and Williams [26] is particularly relevant. A comprehensive survey is available in Wasserman [30].

Feed-forward neural networks such as the network represented in Figure 3 are non-recurrent [30] because outputs do not feedback to inputs. Feed-forward networks are known to be unconditionally stable. In contrast, the feedbacks that characterize recurrent networks make such networks dynamic. Recurrent networks have considerable potential for data compression and pattern recognition, because a trajectory of network outputs is available from a single input [14, 30]. For example, Kosko’s Bidirectional Associative Memory [17] is a recurrent network structure that recognizes patterns by using estimated outputs to refine inputs, and then feeding back the refined inputs to improve the estimated outputs.

Unfortunately, recurrent networks may also be unstable. A sufficient but not a necessary condition for recurrent network stability has been defined by Cohen and Grossberg [4]. However, this condition is narrow, and the outputs of a recurrent neural network cannot generally be assumed to converge. Our research incorporates only nonrecurrent neural networks, though our recurrent associative memory applications incorporate concepts associated with recurrent neural network structure.

3. THE SOLOW-SWAN MODEL OF MACROECONOMIC GROWTH

The Solow-Swan model of macroeconomic growth [12, 24, 25] has been the subject of considerable empirical and theoretical investigation. It is an elegant but highly representative example of a dynamic economic system. Define \( L(t) \) to be participating labor force at time \( t \), and \( K(t) \) to be capital used in production at time \( t \). Let the output of the economy at time \( t \) be described by a Cobb-Douglas production function with constant returns to scale,

\[
Y(t) = f(K(t), L(t)) = Q_0 \cdot [K^\theta \cdot L^{(1-\theta)}] \tag{16}
\]

where \( Q_0 \) and \( \theta \) are constants. Choosing units for \( K(t) \) and \( L(t) \) such that

\[
Q_0 = 1 \tag{17}
\]

let the rate of change in the participating labor force be described by
\[ \frac{\partial L(t)}{\partial t} = \lambda \cdot L(t) \]  \hspace{1cm} (18)

where

\[ \lambda \geq 0 \]  \hspace{1cm} (19)

is a constant growth rate and

\[ L(0) > 0 \]  \hspace{1cm} (20)

Let the rate of change in the available capital stock be described by

\[ \frac{\partial K(t)}{\partial t} = s \cdot Y(t) - \delta \cdot K(t) \]
\[ = s \cdot \left[ K^d \cdot L^{(1-d)} \right] - \delta \cdot K(t) \]  \hspace{1cm} (21)

where \( s \) is a constant savings proportion of national product at time \( t \), and \( \delta \) is a constant depreciation proportion of capital stock at time \( t \). Both \( s \) and \( \delta \) lie on the unit interval \([0, 1]\). Define the capital-labor ratio at time \( t \) to be

\[ k(t) = \frac{K(t)}{L(t)} \]  \hspace{1cm} (22)

Since

\[ K(t) = k(t) \cdot L(t) \]  \hspace{1cm} (23)

it follows that:

\[ \frac{\partial K(t)}{\partial t} = [\frac{\partial k(t)}{\partial t}] \cdot L(t) + k(t) \cdot \left[ \frac{\partial L(t)}{\partial t} \right] \]  \hspace{1cm} (24)

![A Feed-Forward Neural Network with One Hidden Layer](image)

**Fig. 3** A Feed-Forward Neural Network with One Hidden Layer: \( w_{i,j,k} \) is the connective strength to be multiplied by the signal passed from neuron \( i \) in layer \( j \) to neuron \( k \) in layer \( j+1 \).
Substituting equation (18) into equation (24), equating the result with expression (21), and dividing through by $L(t)$,

$$s \cdot [k(t)]^\delta - \delta \cdot k(t) = s \cdot [k(t)]^\theta$$

(25)

Rearranging terms reveals that the time path of the capital-labor ratio for the Solow-Swan growth model is described by a Bernoulli nonlinear ordinary differential equation of the form

$$\frac{\partial k(t)}{\partial t} + (\lambda + \delta) \cdot k(t) = s \cdot [k(t)]^\theta$$

(26)

The solution to this nonlinear ordinary differential equation is easily obtained by defining

$$m(t) = [k(t)]^{(1-\theta)}$$

(27)

and solving the resulting linear ordinary differential equation for the closed form of $m(t)$. Expressed in terms of $k(t)$, the closed form solution is

$$k(t) = \left( [k(0)]^{(1-\theta)} - \frac{s}{(\delta + \lambda)} \right) \cdot \text{EXP}[-(\delta + \lambda) \cdot (1 - \theta) \cdot t] + \left[ \frac{s}{(\delta + \lambda)} \right]^{1/(1-\theta)}$$

(28)

Given the sign conditions imposed on $s$, $\delta$, $\lambda$ and $\theta$,

$$\lim_{t \to \infty} k(t) = \left[ \frac{s}{(\delta + \lambda)} \right]^{1/(1-\theta)}$$

(29)

It is a simple matter to evaluate $k(t)$ if the parameters $s$, $\delta$, $\lambda$, $\theta$, and the initial condition $k(0)$ are known. However, the inverse problem is much more difficult. Given a $p$-dimensional vector $[k(t_1), k(t_2), ..., k(t_p)]$, what are the system parameters that define this sequence of system states? In general, an observed system time path might include noise. In any event, the standard approach is to formulate the inverse problem as a nonlinear least squares problem in which the quantity

$$Z = \sum_{i=1}^{p} \left( k(t_i) - \left( [k(0)]^{(1-\theta)} - \frac{s}{(\delta + \lambda)} \right) \cdot \text{EXP}[-(\delta + \lambda) \cdot (1 - \theta) \cdot t] + \left[ \frac{s}{(\delta + \lambda)} \right]^{1/(1-\theta)} \right)^2$$

(30)

is minimized over the parameter vector $[s, \delta, \lambda, \theta, k(0)]$. As noted in §1, a number of convergent methods can be used to identify the maximally likely values of these parameters, but only if good initial estimates are available.

4. NUMERICAL APPLICATIONS

Kalaba and Tesfatsion [12] used several versions of a multicriteria associative memory to generate training estimates of the parameter values for a set of Solow-Swan growth trajectories. Their investigation included deterministic trajectories, and time paths perturbed by correlated and uncorrelated noise. They found multicriteria associative memories to be very robust, providing good parameter estimates in both the deterministic and stochastic cases. For the sake of comparison and to help direct further research, we generate competing parameter estimates for Kalaba and Tesfatsion’s deterministic systems. We proceed by applying a recurrent associative memory, and by back-propagating errors to determine the connective strengths for a Rumelhart [26, 27] feed-forward neural network.

4.1 Training Estimates of Solow-Swan Parameters
Kalaba and Tesfatsion [12] combined the following parameter values to define training trajectories for their multicriteria associative memories,

\[ \lambda = 0.03, \]  
(31)

\[ \theta = (0.20, 0.23, 0.26, 0.29, 0.32, 0.35, 0.38), \]  
(32)

\[ \delta = 0.07, \]  
(33)

\[ s = (0.9, 0.11, 0.13, 0.15, 0.17, 0.19, 0.21), \]  
(34)

and

\[ k(0) = (4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0). \]  
(35)

Their research relied, in part, on two deterministic training sets. Training set I was created by setting

\[ s = 0.15, \]  
(36)

and combinatorializing the training values of \( k(0) \) and \( \theta \) to form a grid of 49 training responses, each response consisting of a parameter vector of length two. Applying equation (28), these define 49 trajectories for the capital-labor ratio \( k(t) \). Discretizing time over the vector

\[ t = (0.05, 1.05, \ldots, 14.05), \]  
(37)

these system time paths define training stimuli of length 15. Similarly, training set II was created by setting

\[ k(0) = 5.0, \]  
(38)

and combinatorializing the training values for \( \theta \) and \( s \) to produce a second set of 49 stimulus-response pairs.

Representative time paths from training set I are graphed in Figure 4. The value of the initial condition \( k(0) \) has substantial influence on the shape of the path. On the assumption that it is more difficult to estimate parameters for time paths initiated closer the asymptote, we extend Kalaba and Tesfatsion's training data by defining training set III. We set

\[ k(0) = 0.29; \]  
(39)

and, as in the case of training set II, combinatorialize values of \( \theta \) and \( s \). This value of \( k(0) \) is more than an order of magnitude smaller than the values selected by Kalaba and Tesfatsion. Further, it is the median value of \( \theta \). We assume that this selection contributes ambiguity to the parameter identification problem.

4.1.1 Training with multicriteria associative memories

As noted in §2, the multicriteria approach is premised on the introduction of \( \alpha \), which defines a trade-off between the quality of the training approximations generated by applying \( M^\lambda \) and the magnitudes of the elements in \( M^\lambda \). Colby and Tesfatsion [13] searched the interval \([0.1, 1.0]\) for the value of \( \alpha \) that provided the most accurate estimates of the training parameters. They evaluated the quality of their training estimates in terms of percent error,
\begin{equation}
\text{dist}_{k^j}(\alpha) = \{[r_{k^j}(\alpha)^\wedge - r_{k^j}] / r_{k^j}\} \cdot 100
\end{equation}

where \( r_{k^j} \) is the parameter constituting the \( j^{\text{th}} \) element of response vector \( k \), and \( r_{k^j}(\alpha)^\wedge \) is the multicriteria associative memory estimate of \( r_{k^j} \) using trade-off coefficient value \( \alpha \). In the deterministic case, Kalaba and Tesfatsion’s best parameter estimates were associated with the value

\begin{equation}
\alpha = 0.90
\end{equation}

![Graph showing representative time paths for the Solow-Swan Growth Model: Training set I.](image)

Fig. 4  Representative Time Paths for the Solow-Swan Growth Model: Training set I.

Surprisingly, the quality of the estimates associated with

\begin{equation}
\alpha = 0.10
\end{equation}

were almost as good as the estimates computed under (41). Setting

\begin{equation}
\alpha = 1.0
\end{equation}

eliminates the trade-off between the \( L_2 \) norms of \( R - M^\wedge \cdot S \) and \( M^\wedge \), and provided Kalaba and Tesfatsion’s worst case parameter estimates. The difference between the best and worst case performances was considerable. Attaching the small relative penalty of

\begin{equation}
1 - \alpha = 0.1
\end{equation}
to the $L_2$ norm of $M^\top$ reduced the relative error in the associated parameter estimates by two to three orders of magnitude relative to the worst case. Attaching any penalty on the $\alpha$ interval $[0.1, 1)$ provides improvement relative to the results generated with equation (43).

**Fig. 5a** Training Set I, $s = 0.15$: (i.) Percent error in the Multicriteria Associative Memory (MAM) training estimates of $k(0)$ as a function of $k(0)$ and $\theta$. (ii.) Percent error in the MAM training estimates of $\theta$ as a function of $\theta$ and $k(0)$.

**Fig. 5b** Training Set II, $k(0) = 5.0$: (iii.) Percent error in the Multicriteria Associative Memory (MAM) training estimates of $s$ as a function of $s$ and $\theta$. (iv.) Percent error in the MAM training estimates of $\theta$ as a function of $\theta$ and $s$.

**Fig. 5c** Training Set III, $k(0) = 0.29$: (v.) Percent error in the Multicriteria Associative Memory (MAM) training estimates of $s$ as a function of $s$ and $\theta$. (vi.) Percent error in the MAM training estimates of $\theta$ as a function of $\theta$ and $s$. 

99
The surfaces in Figures 5a,b summarize the percent errors in Kalaba and Tesfatsion's most accurate multicriteria associative memory estimates. Figure 5c provides the same information for estimates provided by training set III. In each case, percent errors are plotted as a function of the true parameter values defining the training grid. The error surfaces are generated by the method of distance weighted least squares [18, 31]. Unlike polynomial smoothing procedures, distance weighted least squares produces a surface that flexes locally to fit the data. These interpolative surfaces are subjected to modest extrapolations near the edges of the training grid.

The multicriteria approach provided highly accurate training estimates of the initial capital-labor ratio \( k(0) \). Kalaba and Tesfatsion reported relative estimation errors of less than one percent, and thus the corresponding percent error surface has no discernible curvature. Training set I and II estimates of \( s \) and \( \theta \) are less good, but the corresponding percent estimation errors rarely exceed 20 percent. As expected, the parameter estimates generated with training set III are the least accurate, but even these are serviceable if the error surface is not extrapolated beyond the training grid.

### 4.1.2 Training with recurrent associative memories

Percent error surfaces for our recurrent associative memory estimates appear in Figures 6a,b,c. These correspond respectively to the percent error surfaces for the multicriteria associative memory estimates summarized in Figures 5a,b,c. The recurrent training estimates are consistently more accurate than Kalaba and Tesfatsion's [13] corresponding multi criteria estimates. As was the case in Figure 5a, the estimation error associated with the initial condition \( k(0) \) is negligible. The vertical scales in Figures 6a,b,c have been adjusted relative to Figures 5a,b,c to make the relative errors in the recurrent associative memory estimates more apparent. Without scale adjustments, the error surfaces would reveal very little curvature, in part because the recurrent estimation errors are much less systematically distributed over the training grid than are the multicriteria errors. Further, though the orthogonal arrangement of axes is consistent across Figures 5a,b,c and Figures 6a,b,c, direction of increase is manipulated to provide maximum visual access to the relative error surfaces.

With the exception of case (vi), the estimates of \( \theta \) generated with the training set III, we argue that further attempts to refine the recurrent training estimates are unnecessary. These estimates are of such high quality that we expect convergent algorithms could do little to improve them.

### 4.1.3 Training with feed-forward neural networks

Multicriteria and recurrent associative memories both provide useful estimates of training parameters, but recovering training parameters is not a particularly useful exercise. Training parameters are known. If associative memories can provide useful initial estimates for parameter identification problems, then these estimates must be provided based on time paths or other system trajectories not explicitly represented in the training set.

Training a feed-forward neural network via back propagation often requires ad hoc interventions in the training process. In particular, the rate of change in the matrix of connective strengths must often be varied as the training response estimates begin to converge. Given sufficient training time and techniques for avoiding network paralysis, any sufficiently complex neural network can be trained to associate arbitrarily accurate response estimates with training stimuli. However, the utility of a neural network does not lie in its capacity to estimate training outputs. Connectionist models such as neural networks are premised on the prospect of generalizability [30]. The capacity of networks to generalize, and to return good response estimates for previously unobserved stimuli has led to extensive research.

Our interest in applying artificial neural networks to the initial estimates problem
reflects the difficulty of the problem. Connectionist approaches are justified when conventional algorithmic procedures do not exist or are prohibitively expensive. We have no satisfactory algorithm for generating initial parameter estimates for nonlinear systems. As noted above, there are inexpensive algorithms available for refining initial parameter estimates, but producing initial estimates is most often treated as an exercise in expert judgment. In this context, artificial intelligence approaches may eventually prove to be the best procedures available.

Fig. 6a  Training Set I, s = 0.15: (i) Percent error in the Recurrent Associative Memory (RAM) training estimates of k(0) as a function of k(0) and θ. (ii) Percent error in the RAM training estimates of θ as a function of θ and k(0).

Fig. 6b  Training Set II, k(0) = 5.0: (iii) Percent error in the Recurrent Associative Memory (RAM) training estimates of s as a function of s and θ. (iv) Percent error in the RAM training estimates of θ as a function of θ and s.

Fig. 6c  Training Set III, k(0) = 0.29: (v) Percent error in the Recurrent Associative Memory (RAM) training estimates of s as a function of s and θ. (vi) Percent error in the RAM training estimates of θ as a function of θ and s.
A separate neural network is trained on each of the three training sets. The networks are represented in Figure 7. The dimensions of the input and output layers for the network conform to the dimensions of the stimulus and response vectors. In addition, the networks each have one hidden layer consisting of 20 neurons. Figure 7 differs from the archetypical neural network portrayed in Figure 3 only in that the network in Figure 7 includes trainable bias neurons. Bias neurons always return an output value of one, but their connective strengths are subject to training adjustments. Despite these additional adjustments, bias neurons usually reduce training time.

![Diagram of a neural network](image)

**Fig. 7** A Feed-Forward Neural Network Trained to Map Solow-Swan Time Paths to System Parameters: Training response vectors for network I consist of \([k(0), \theta]\)ls = .15. Training response vectors for network II consist of \([s, \theta]\)k(0) = 5.0. Training response vectors for network III consist of \([s, \theta]\)k(0) = .29.

The neural networks for training sets I, II, and III are specified and trained to a five percent (vector) error tolerance in approximately two hours using the GENESIS program [6], a back propagation [26] training algorithm executed on the Delta II floating point
processor [29]. Unfortunately, we are unable to complete the training procedure for the neural network dedicated to training set II. The connective strengths do not converge despite learning rate modifications in the training program. The connective strengths for trained networks I and III are listed in Appendix I.

Percent error surfaces for the neural network training estimates appear in Figures 8a,b. The surfaces correspond to the multicriteria associative memory estimates summarized in Figures 5a,c and to the recurrent associative memory estimates summarized in Figures 6a,c.

Fig. 8a  Training Set I, \( s = 0.15 \): (i.) Percent error in the neural network training estimates of \( k(0) \) as a function of \( k(0) \) and \( \theta \). (ii.) Percent error in the neural network training estimates of \( \theta \) as a function of \( k(0) \) and \( \theta \).

Fig. 8b  Training Set III, \( k(0) = 0.29 \): (iii.) Percent error in the neural network training estimates of \( s \) as a function of \( s \) and \( \theta \). (iv.) Percent error in the neural network training estimates of \( \theta \) as a function of \( \theta \) and \( s \).

The relative errors in with the neural network training estimates vary as systematically as the errors in the multicriteria training estimates, but the neural network error surfaces are more complicated. Overall, the training estimates provided by the neural
networks are more accurate than the multicriteria training estimates, but less accurate than the estimates provided by the recurrent associative memories. Estimates of the initial condition \( k(0) \) are an exception. The multicriteria and recurrent estimates of \( k(0) \) are considerably more accurate than the corresponding neural network estimates. However, as noted above, the comparison between neural network and associative memory training estimates is of small importance. The quality of the training estimates provided by the neural network is a function of network training time. If the vector error tolerance used as the stopping criterion for training is reduced, training will continue. The connective weights in the neural network would be further adjusted, and the quality of the network's training estimates might exceed that of the parameter estimates generated by the recurrent associative memory. The arbitrarily high quality of the neural network training estimates makes clear the importance of testing comparisons based on stimulus/response pairs not used in training.

**Fig. 9a** Training and Test Sets I: \( s = 0.015 \). Training responses are identified by black coordinates. Test responses are white.

**Fig. 9b** Training and Test Sets II and III. \( k(0) = 5.0 \) and 0.29, respectively. Training responses are identified by black coordinates. Test responses are white.
4.2 Test Estimates of Solow-Swan Parameters

As a basis for comparing the quality of test estimates, we combinatorialize across ten values for each parameter. Each value is drawn from a uniform probability distribution over each of the training intervals. These 200 test parameter combinations are summarized in Figures 9a,b. Each test coordinate constitutes a true test response vector. Each coordinate generates a time path that is a test stimulus vector.

Fig. 10a  Test Set I: (i.) Percent error in the Multicriteria Associative Memory (MAM) test estimates of k(0) as a function of k(0) and $\theta$. (ii.) Percent error in the MAM test estimates of $\theta$ as a function of $\theta$ and k(0).

Fig. 10b  Test Set II: (iii.) Percent error in the Multicriteria Associative Memory (MAM) test estimates of s as a function of s and $\theta$. (iv.) Percent error of the MAM test estimates of $\theta$ as a function of $\theta$ and s.

Fig. 10c  Test Set III: (v.) Percent error in the Multicriteria Associative Memory (MAM) test estimates of s as a function of s and $\theta$. (vi.) Percent error of the MAM test estimates of $\theta$ as a function of $\theta$ and s.
4.2.1 Testing with multicriteria associative memories

The surfaces in Figures 10a,b,c summarize the percent errors in the multicriteria associative memory test estimates. As always, percent errors are plotted as a function of the true parameter values. The processes generating the multicriteria training and test surfaces obviously covary. As is true for the training estimates, the multicriteria approach provided the most accurate test estimates in the case of the initial capital-labor ratio k(0). Restricting attention from the fringe of the percent error surfaces that extrapolate the training grid, other test estimates range from fair to good. The test estimates of θ exhibit the greatest degree of degradation relative to the training estimates, but even these estimates are serviceable.

4.2.2 Testing with recurrent associative memories

Percent error surfaces for our recurrent associative memory test estimates appear in Figures 11a,b,c. As is true of the training estimates, the recurrent test estimates are consistently more accurate than the corresponding multicriteria test estimates. As before, the smallest relative estimation error is associated with k(0). In contrast to the multicriteria results, the relative error surfaces plotted for the recurrent testing estimates are often less steep than the error surfaces plotted for the corresponding training estimates. However, the recurrent test estimates demonstrate a greater tendency to consistently under or over predict the true parameters values. This is true relative to both the recurrent training estimates and the multicriteria estimates. A potentially useful interpretation is that the recurrent test estimates are behaving in a fashion similar to ridge estimators, trading off small degrees of bias against a large reduction in visions.

Fig. 11a Test Set I: (i.) Percent error in the Recurrent Associative Memory (RAM) test estimates of k(0) as a function of k(0) and θ. (ii.) Percent error in the RAM test estimates of θ as a function of θ and k(0).

Fig. 11b Test Set II: (iii.) Percent error in the Recurrent Associative Memory (RAM) test estimates of θ as a function of θ and s. (iv.) Percent error in the RAM test estimates of θ as a function of θ and s.
4.2.3 Testing with feed-forward neural networks

Percent error surfaces for the neural network test estimates appear in Figures 12a, b. Unlike the associative memory exercises, the quality of the neural network test estimates contrasts sharply with that of the training estimates. There is so much variance in the neural network test estimates that the error coordinates in Figures 12a,b almost defy interpolation. The distance weighted least squares procedure applied to the previous estimates is computationally very intensive in this case. Consequently, the surfaces in Figures 12a,b cannot flex sufficiently to include every error coordinate. Still, the surfaces are close approximations.

Fig. 11c  Test Set III: (v.) Percent error in the Recurrent Associative Memory (RAM) test estimates of $s$ as a function of $s$ and $8$.  (vi.) Percent error in the RAM test estimates of $8$ as a function of $8$ and $s$.

Fig. 12a  Test Set I, s = 0.15: (i.) Percent error in the neural network test estimates of $k(0)$ as a function of $k(0)$ and $8$. (ii.) Percent error in the neural network test estimates of $8$ as a function of $k(0)$ and $8$. 

107
With few exceptions, the neural network test estimates are not accurate enough to use as initial parameter estimates. Though unimpressive relative to the associative memory estimates, the percent errors in the neural network estimates are lower than the relative errors we have encountered in other neural network applications [20]. We assume that this improvement is related to the relatively long time paths used to define the stimulus vectors.

![Graphs showing percent error in neural network estimates]

**Fig. 12b** Test Set III, $k(0) = 0.29$: (iii.) Percent error in the neural network test estimates of $s$ as a function of $s$ and $\theta$. (iv.) Percent error in the neural network test estimates of $\theta$ as a function of $\theta$ and $s$.

If the response vectors are substantially longer than the stimulus vectors, then the probability of a non-unique vector mapping is increased. In this case, the response vectors are obviously quite short relative to the stimuli. The possibility of inconsistent training data remains important however, because any time parameter vector for which

$$k(0) = \left[ s / (\delta + \lambda) \right]^{1/(1-\theta)}$$

(45)

implies a horizontal time path. In this exercise, we are careful to exclude the asymptote from the various training and test sets, and thus avoid inconsistent data.

If the training grid were further constrained, the quality of the neural network test estimates would probably improve. However, narrowing the training grid or reducing the response vector to a scalar would greatly simplify the problem. The recurrent associative memory matrices perform so well in the current context, that additional investments in neural network training are unattractive.

5. CONCLUSIONS AND EXTENSIONS

Our recurrent associative memories provide more accurate parameter estimates for the Solow-Swan macroeconomic growth model than do Kalaba and Tesfatsion's [12] multi-criteria associative memories, or our feed-forward neural network. The recurrent associative memories consistently provide the best response estimates to both training and test stimuli. Despite training performance comparable to that of the associative memories, the neural network does not provide useful test estimates of system parameters. This is surprising, given the neural network literature's emphasis on generalization.

We know from previous work that recurrent associative memories perform well in deterministic contexts [7, 8, 20], but we also have computational evidence that recurrent associative memories are very sensitive to noise [21]. In this investigation, the time
paths defined by the Bernoulli differential equation underlying the Solow-Swan model are deterministic. Unlike Kalaba and Tesfatsion, we do not consider perturbing the stimulus vectors with noise. If we widen the comparison to include stochastic versions of the Solow-Swan trajectories, we expect that the quality of the recurrent associative memory estimates would decline significantly. In the presence of noise, the multicriteria associative memories might well outperform the recurrent extension.

![Diagram](image)

**Fig. 13** Computing and Applying a Multicriteria Recurrent Associative Memory (MRAM) Matrix $M^{*o}$. 

1. **Initialization (Training)**
   - Given a population of paired stimulus vectors $s$ and response vectors $r$, identify a representative training set of stimulus-response pairs.

2. Group the elements of the training set to form the training stimulus matrix $S$ and the training response matrix $R$.

3. Form the multicriteria associative memory matrix $M^* = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot (I - ad)^{-1}$.

4. Estimate the training response vector $r$ from the corresponding stimulus vectors $s$ as $R^* = M^* \cdot S$.

5. Recurrence
   - Form the extended test stimulus vectors $S$.

6. Reestimate the training response vectors $r$ from the corresponding extended stimulus vectors $s$ as $R^* = M^* \cdot (R + S)$.

Start (Train).

1. **Initialization (Testing)**
   - Given a population of stimulus vectors $s$, identify a representative set of test stimuli. The corresponding response vectors $r$ are unknown.

2. Group the elements of the testing set to form the test stimulus matrix $S$. The corresponding test response matrix $R$ is unknown.

3. Given the multicriteria associative memory matrix $M^*$, estimate the test response vectors $r$ from corresponding stimulus vectors $s$ as $R^* = M^* \cdot S$.

4. Form the extended test stimulus vectors $S$.

5. Given the multicriteria recurrent associative memory matrix $M^{*o}$, reestimate the test response vectors $r$ from corresponding extended stimulus vectors $s$ as $R^* = M^{*o} \cdot (R + S)$.

Stop with $R^*$ (testing).
There is no compelling reason why the recurrent and associative memory approaches cannot be combined. Computing a multicriteria associative memory limits potential ill-conditioning of the memory matrix elements. Recurrent associative memories are as subject to ill-conditioning as linear associative memories. Further, computational experience indicates that the problems associated with ill-conditioning are exacerbated in the recurrent case. A trade-off can be introduced between the quality of the approximation provided by a recurrent associative memory and the size of the elements in the memory matrix. Hopefully, this approach will reduce the recurrent procedure’s sensitivity to noise while retaining the capacity to produce very good initial parameter estimates. Combining equations (11) and (14) produces

\[
M^{**} = \alpha \cdot R \cdot \left[ \frac{S}{f(R^*)} \right]^T \cdot \left[ \alpha \cdot \left[ \frac{S}{f(R^*)} \right] \right]^T + (1 - \alpha) \cdot I \right]^{-1} \tag{46}
\]

An algorithm for computing such a multicriteria recurrent associative memory (MRAM) matrix appears in Figure 13. The multicriteria trade-off might be introduced in step 3 of the training procedure, in step 5, or in both. Computational and theoretical research is currently underway at the University of Southern California concerning

- the feasibility of using a hybrid, multicriteria/recurrent approach for obtaining initial parameter estimates for stochastic nonlinear systems, and
- a general explanation of why our interpretation of recurrence performs so well in this and other deterministic contexts.

REFERENCES


5. J. Hopfield and D. Tank, "Neural" computation of decisions in optimization problems, Biological Cybernetics 52 (1985) 141-152.


29. Science International Applications Corporation, Delta II 22 MFLOP PC accelerator board (Artificial Neural Systems Division, 10260 Campus Point Dr., MS 71, San Diego, CA 92121, 1990).


Acknowledgment

The authors gratefully acknowledge the technical assistance and advice of Gary Josin, PhD, and Doug Chaney, PhD, Neural Systems Incorporated, Vancouver. The usual caveat applies.

---

**APPENDIX**

**Logistic Transfer Function**

Defining $x_{i,j,k}$ to be the signal passed from unit $i$ in layer $j$ to unit $k$ in layer $j+1$, and $w_{i,j,k}$ to be the connective strength between unit $i$ in layer $j$ and unit $k$ in layer $j+1$, the logistic transfer function is

$$x_{k,j+1,l} = \left\{ 1 + \exp\left(-\left(\Sigma_i w_{i,j,k} x_{i,j,k} + b_{j+1}\right)\right) \right\}^{-1} - 0.5. \quad (46)$$

**Bias and Connective Strengths for a Trained Neural Network with 15 Units in the Input Layer, 20 Units in the Hidden Layer, and 2 Units in the Output Layer**

Training Set I: $r = \{k(0), \theta\}$, $s = .15$

<table>
<thead>
<tr>
<th>INPUT LAYER</th>
<th>UNIT 1</th>
<th>UNIT 2</th>
<th>UNIT 3</th>
<th>UNIT 4</th>
<th>UNIT 5</th>
<th>UNIT 6</th>
<th>UNIT 7</th>
<th>UNIT 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT 1</td>
<td>7.5908</td>
<td>1.9120</td>
<td>0.0070</td>
<td>-0.0841</td>
<td>-13.5664</td>
<td>-0.0139</td>
<td>0.0740</td>
<td>0.0356</td>
</tr>
<tr>
<td>UNIT 2</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0132</td>
<td>-0.0000</td>
<td>-0.0098</td>
<td>0.0739</td>
<td>0.0146</td>
</tr>
<tr>
<td>UNIT 3</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>-6.1217</td>
<td>-0.0113</td>
<td>0.0009</td>
<td>0.0144</td>
</tr>
<tr>
<td>UNIT 4</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>0.0009</td>
<td>-0.0127</td>
<td>0.0108</td>
<td>0.0586</td>
</tr>
<tr>
<td>UNIT 5</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0113</td>
<td>-0.0828</td>
<td>0.0232</td>
</tr>
<tr>
<td>UNIT 6</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0131</td>
<td>0.0102</td>
<td>-0.0381</td>
</tr>
<tr>
<td>UNIT 7</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>4.5768</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0052</td>
</tr>
<tr>
<td>UNIT 8</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>0.0031</td>
<td>-0.0223</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>UNIT 9</td>
<td>0.0002</td>
<td>0.0000</td>
<td>0.0128</td>
<td>0.0000</td>
<td>0.0013</td>
<td>0.0248</td>
<td>-0.1684</td>
<td>0.0000</td>
</tr>
<tr>
<td>UNIT 10</td>
<td>0.0000</td>
<td>0.0001</td>
<td>-0.0001</td>
<td>0.0047</td>
<td>9.5417</td>
<td>0.0527</td>
<td>-0.1653</td>
<td>-0.0392</td>
</tr>
<tr>
<td>UNIT 11</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0124</td>
<td>10.3684</td>
<td>-0.0200</td>
<td>-0.2063</td>
<td>-0.0820</td>
</tr>
<tr>
<td>UNIT 12</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>11.5132</td>
<td>0.0340</td>
<td>-0.1776</td>
<td>-0.0448</td>
</tr>
<tr>
<td>UNIT 13</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>0.0005</td>
<td>0.0338</td>
<td>-0.2277</td>
<td>-0.0253</td>
</tr>
<tr>
<td>UNIT 14</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0019</td>
<td>-0.1697</td>
<td>-0.0569</td>
</tr>
<tr>
<td>UNIT 15</td>
<td>0.0469</td>
<td>0.2218</td>
<td>-0.0163</td>
<td>-0.0776</td>
<td>13.6100</td>
<td>0.0646</td>
<td>-0.1563</td>
<td>-0.0257</td>
</tr>
<tr>
<td>BIAS</td>
<td>-4.7317</td>
<td>-0.7280</td>
<td>-0.0127</td>
<td>0.0980</td>
<td>16.2581</td>
<td>-0.0931</td>
<td>0.4270</td>
<td>0.3064</td>
</tr>
</tbody>
</table>
### HIDDEN LAYER

<table>
<thead>
<tr>
<th>INPUT LAYER</th>
<th>UNIT 9</th>
<th>UNIT 10</th>
<th>UNIT 11</th>
<th>UNIT 12</th>
<th>UNIT 13</th>
<th>UNIT 14</th>
<th>UNIT 15</th>
<th>UNIT 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT 1</td>
<td>-1.5369</td>
<td>-0.2025</td>
<td>1.1601</td>
<td>-0.3169</td>
<td>1.7371</td>
<td>0.0405</td>
<td>0.1791</td>
<td>0.3325</td>
</tr>
<tr>
<td>UNIT 2</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>0.2012</td>
<td>-0.2263</td>
<td>1.5201</td>
<td>0.0577</td>
<td>0.0890</td>
<td>0.3743</td>
</tr>
<tr>
<td>UNIT 3</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0261</td>
<td>1.3199</td>
<td>-0.0015</td>
<td>0.0754</td>
<td>0.3652</td>
</tr>
<tr>
<td>UNIT 4</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>1.0951</td>
<td>0.0968</td>
<td>0.0896</td>
<td>0.2956</td>
</tr>
<tr>
<td>UNIT 5</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>0.0001</td>
<td>0.0717</td>
<td>-0.0231</td>
<td>0.2854</td>
</tr>
<tr>
<td>UNIT 6</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.2662</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0314</td>
<td>0.3519</td>
</tr>
<tr>
<td>UNIT 7</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>0.3678</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.3357</td>
</tr>
<tr>
<td>UNIT 8</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>-0.2120</td>
<td>0.4354</td>
<td>-0.0000</td>
<td>0.1189</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>UNIT 9</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-1.0099</td>
<td>0.4696</td>
<td>0.0003</td>
<td>0.1045</td>
<td>-0.0633</td>
<td>0.0000</td>
</tr>
<tr>
<td>UNIT 10</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-1.1312</td>
<td>0.5711</td>
<td>0.4825</td>
<td>0.1301</td>
<td>-0.1275</td>
<td>0.2945</td>
</tr>
<tr>
<td>UNIT 11</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.5863</td>
<td>0.3686</td>
<td>0.1387</td>
<td>-0.1430</td>
<td>0.2815</td>
</tr>
<tr>
<td>UNIT 12</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.3360</td>
<td>0.0969</td>
<td>-0.1241</td>
<td>0.2213</td>
</tr>
<tr>
<td>UNIT 13</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>-1.3947</td>
<td>0.1295</td>
<td>-0.0000</td>
<td>0.1750</td>
<td>-0.1832</td>
<td>0.2721</td>
</tr>
<tr>
<td>UNIT 14</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>-1.4963</td>
<td>0.6953</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.2108</td>
<td>0.2463</td>
</tr>
<tr>
<td>UNIT 15</td>
<td>1.0587</td>
<td>0.0149</td>
<td>-1.4981</td>
<td>0.7228</td>
<td>0.2011</td>
<td>0.1836</td>
<td>-0.1422</td>
<td>0.2455</td>
</tr>
<tr>
<td>BIAS</td>
<td>0.1648</td>
<td>0.1476</td>
<td>-1.7083</td>
<td>-0.29207</td>
<td>-0.9824</td>
<td>-0.4887</td>
<td>0.2102</td>
<td>0.5822</td>
</tr>
</tbody>
</table>

### OUTPUT LAYER

<table>
<thead>
<tr>
<th>INPUT LAYER</th>
<th>UNIT 17</th>
<th>UNIT 18</th>
<th>UNIT 19</th>
<th>UNIT 20</th>
<th>HIDDEN LAYER</th>
<th>UNIT 1</th>
<th>UNIT 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT 1</td>
<td>0.9403</td>
<td>0.4642</td>
<td>-5.5635</td>
<td>-7.9748</td>
<td>UNIT 1</td>
<td>10.5690</td>
<td>0.0357</td>
</tr>
<tr>
<td>UNIT 2</td>
<td>0.9177</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0012</td>
<td>UNIT 2</td>
<td>0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 3</td>
<td>0.9620</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>UNIT 3</td>
<td>0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 4</td>
<td>0.9227</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>UNIT 4</td>
<td>0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 5</td>
<td>0.8567</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>UNIT 5</td>
<td>0.0000</td>
<td>6.8101</td>
</tr>
<tr>
<td>UNIT 6</td>
<td>0.8244</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>UNIT 6</td>
<td>0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 7</td>
<td>0.7859</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>UNIT 7</td>
<td>0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 8</td>
<td>0.7927</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>UNIT 8</td>
<td>0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 9</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>UNIT 9</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>UNIT 10</td>
<td>0.0000</td>
<td>0.0003</td>
<td>0.0000</td>
<td>0.0152</td>
<td>UNIT 10</td>
<td>0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 11</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0001</td>
<td>UNIT 11</td>
<td>-0.0000</td>
<td>-0.0235</td>
</tr>
<tr>
<td>UNIT 12</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>UNIT 12</td>
<td>-0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>UNIT 13</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>UNIT 13</td>
<td>0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 14</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>UNIT 14</td>
<td>0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 15</td>
<td>0.6506</td>
<td>-0.3940</td>
<td>16.6467</td>
<td>2.0000</td>
<td>UNIT 15</td>
<td>0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>BIAS</td>
<td>0.5598</td>
<td>-0.1023</td>
<td>5.4366</td>
<td>-0.3776</td>
<td>BIAS</td>
<td>1.1759</td>
<td>-1.2480</td>
</tr>
</tbody>
</table>

| UNIT 16     | 0.0000  | 0.0000  | 0.0000  | 0.0000  | UNIT 16     | 0.0000| -0.0000 |
| UNIT 17     | 0.0000  | 0.0000  | 0.0000  | 0.0000  | UNIT 17     | 0.0005| -0.0000 |
| UNIT 18     | 0.0000  | 0.0000  | 0.0000  | 0.0000  | UNIT 18     | -0.0000| -0.0001 |
| UNIT 19     | 0.0000  | 0.0000  | 0.0000  | 0.0000  | UNIT 19     | 0.0000| 6.1929  |
### Bias and Connective Strengths for a Trained Neural Network with 15 Units in the Input Layer, 20 Units in the Hidden Layer, and 2 Units in the Output Layer

**Training Set III: r = [s, 8], k(0) = .29**

<table>
<thead>
<tr>
<th>INPUT LAYER</th>
<th>UNIT 1</th>
<th>UNIT 2</th>
<th>UNIT 3</th>
<th>UNIT 4</th>
<th>UNIT 5</th>
<th>UNIT 6</th>
<th>UNIT 7</th>
<th>UNIT 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT 1</td>
<td>-0.0241</td>
<td>0.0149</td>
<td>-0.0192</td>
<td>-0.0400</td>
<td>0.5384</td>
<td>0.0272</td>
<td>0.0020</td>
<td>-0.0221</td>
</tr>
<tr>
<td>UNIT 2</td>
<td>0.0004</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0132</td>
<td>0.0014</td>
<td>-0.0239</td>
<td>0.0303</td>
<td>-0.0257</td>
</tr>
<tr>
<td>UNIT 3</td>
<td>-0.0003</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0085</td>
<td>0.0565</td>
<td>-0.0038</td>
<td>-0.0001</td>
<td>-0.0100</td>
</tr>
<tr>
<td>UNIT 4</td>
<td>-0.0003</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>0.0140</td>
<td>0.0161</td>
<td>0.0500</td>
</tr>
<tr>
<td>UNIT 5</td>
<td>-0.0003</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0120</td>
<td>-0.0565</td>
<td>0.0280</td>
</tr>
<tr>
<td>UNIT 6</td>
<td>-0.0003</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0549</td>
<td>-0.0202</td>
</tr>
<tr>
<td>UNIT 7</td>
<td>-0.0002</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.2513</td>
<td>-0.0000</td>
<td>0.0347</td>
</tr>
<tr>
<td>UNIT 8</td>
<td>-0.0002</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0436</td>
<td>-0.0072</td>
<td>-0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 9</td>
<td>-0.0005</td>
<td>-0.0000</td>
<td>-0.0187</td>
<td>-0.0000</td>
<td>0.0575</td>
<td>-0.0392</td>
<td>-0.0768</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 10</td>
<td>0.0001</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0464</td>
<td>1.0914</td>
<td>0.0641</td>
<td>-0.0604</td>
<td>0.0209</td>
</tr>
<tr>
<td>UNIT 11</td>
<td>0.0009</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0022</td>
<td>1.4283</td>
<td>-0.0105</td>
<td>-0.0903</td>
<td>-0.0129</td>
</tr>
<tr>
<td>UNIT 12</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>1.8032</td>
<td>0.0421</td>
<td>-0.0507</td>
<td>0.0324</td>
</tr>
<tr>
<td>UNIT 13</td>
<td>0.0001</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0001</td>
<td>0.0403</td>
<td>-0.0917</td>
<td>0.0596</td>
</tr>
<tr>
<td>UNIT 14</td>
<td>0.0002</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0001</td>
<td>0.0104</td>
<td>-0.0242</td>
<td>0.0347</td>
</tr>
<tr>
<td>UNIT 15</td>
<td>5.8158</td>
<td>0.0590</td>
<td>-0.0126</td>
<td>-0.0337</td>
<td>2.9432</td>
<td>0.0685</td>
<td>-0.0027</td>
<td>0.0724</td>
</tr>
<tr>
<td>BIAS</td>
<td>0.8814</td>
<td>-0.0049</td>
<td>0.0049</td>
<td>-0.0079</td>
<td>-1.1354</td>
<td>-0.0064</td>
<td>-0.0356</td>
<td>0.0086</td>
</tr>
</tbody>
</table>

### HIDDEN LAYER

<table>
<thead>
<tr>
<th>INPUT LAYER</th>
<th>UNIT 9</th>
<th>UNIT 10</th>
<th>UNIT 11</th>
<th>UNIT 12</th>
<th>UNIT 13</th>
<th>UNIT 14</th>
<th>UNIT 15</th>
<th>UNIT 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT 1</td>
<td>0.0110</td>
<td>0.0238</td>
<td>-0.0379</td>
<td>-0.0381</td>
<td>-0.2089</td>
<td>0.0161</td>
<td>-0.1591</td>
<td>-3.6563</td>
</tr>
<tr>
<td>UNIT 2</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0338</td>
<td>-0.0194</td>
<td>0.0500</td>
<td>0.0151</td>
<td>-0.0235</td>
<td>0.0160</td>
</tr>
<tr>
<td>UNIT 3</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.1050</td>
<td>0.0556</td>
<td>-0.0530</td>
<td>0.0526</td>
<td>0.0206</td>
</tr>
<tr>
<td>UNIT 4</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>0.0136</td>
<td>0.0369</td>
<td>0.1407</td>
<td>-0.0350</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 5</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0028</td>
<td>0.0872</td>
<td>-0.0309</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 6</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>0.2033</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.1259</td>
<td>0.0497</td>
</tr>
<tr>
<td>UNIT 7</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0175</td>
<td>0.2464</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0473</td>
</tr>
<tr>
<td>UNIT 8</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0157</td>
<td>0.2555</td>
<td>0.0008</td>
<td>0.0274</td>
<td>-0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 9</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0266</td>
<td>0.2364</td>
<td>0.1034</td>
<td>0.0058</td>
<td>0.1713</td>
<td>0.0000</td>
</tr>
<tr>
<td>UNIT 10</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0161</td>
<td>0.2910</td>
<td>0.1425</td>
<td>0.0247</td>
<td>0.1182</td>
<td>0.0469</td>
</tr>
<tr>
<td>UNIT 11</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>0.2567</td>
<td>0.1070</td>
<td>0.0268</td>
<td>0.1075</td>
<td>0.0471</td>
</tr>
<tr>
<td>UNIT 12</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>0.1433</td>
<td>-0.0211</td>
<td>0.1238</td>
<td>-0.0019</td>
</tr>
<tr>
<td>UNIT 13</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0062</td>
<td>0.2602</td>
<td>0.0000</td>
<td>0.0508</td>
<td>0.0567</td>
<td>0.0622</td>
</tr>
<tr>
<td>UNIT 14</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0368</td>
<td>0.2312</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>0.0173</td>
<td>0.0482</td>
</tr>
<tr>
<td>UNIT 15</td>
<td>-1.2507</td>
<td>0.0587</td>
<td>0.0116</td>
<td>0.2158</td>
<td>0.1685</td>
<td>0.0479</td>
<td>0.0709</td>
<td>0.0597</td>
</tr>
</tbody>
</table>

**BIAS** | -0.0073 | 0.0196  | -0.0265 | 0.2091  | 0.0700  | -0.00812| 0.0892  | 0.0163  |

114
### HIDDEN LAYER

<table>
<thead>
<tr>
<th>INPUT LAYER</th>
<th>UNIT 17</th>
<th>UNIT 18</th>
<th>UNIT 19</th>
<th>UNIT 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT 1</td>
<td>-0.2089</td>
<td>0.0162</td>
<td>-0.1591</td>
<td>-3.6563</td>
</tr>
<tr>
<td>UNIT 2</td>
<td>0.0425</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0049</td>
</tr>
<tr>
<td>UNIT 3</td>
<td>0.3536</td>
<td>0.0000</td>
<td>0.0001</td>
<td>-0.0031</td>
</tr>
<tr>
<td>UNIT 4</td>
<td>0.5636</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0027</td>
</tr>
<tr>
<td>UNIT 5</td>
<td>0.7249</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0027</td>
</tr>
<tr>
<td>UNIT 6</td>
<td>0.9128</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0145</td>
</tr>
<tr>
<td>UNIT 7</td>
<td>1.0645</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0018</td>
</tr>
<tr>
<td>UNIT 8</td>
<td>1.2442</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0200</td>
</tr>
<tr>
<td>UNIT 9</td>
<td>0.6068</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>0.0163</td>
</tr>
<tr>
<td>UNIT 10</td>
<td>-0.0000</td>
<td>0.0000</td>
<td>-0.0000</td>
<td>7.7730</td>
</tr>
<tr>
<td>UNIT 11</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>6.7110</td>
</tr>
<tr>
<td>UNIT 12</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0012</td>
</tr>
<tr>
<td>UNIT 13</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0011</td>
</tr>
<tr>
<td>UNIT 14</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0000</td>
<td>-0.0019</td>
</tr>
<tr>
<td>UNIT 15</td>
<td>1.9330</td>
<td>-0.1031</td>
<td>-2.6518</td>
<td>-6.8646</td>
</tr>
</tbody>
</table>

### OUTPUT LAYER

<table>
<thead>
<tr>
<th>HIDDEN LAYER</th>
<th>UNIT 1</th>
<th>UNIT 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT 1</td>
<td>3.2632</td>
<td>23.9190</td>
</tr>
<tr>
<td>UNIT 2</td>
<td>0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 3</td>
<td>-0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>UNIT 4</td>
<td>-0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>UNIT 5</td>
<td>0.0000</td>
<td>1.3113</td>
</tr>
<tr>
<td>UNIT 6</td>
<td>0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 7</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>UNIT 8</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>UNIT 9</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>UNIT 10</td>
<td>-0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 11</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>UNIT 12</td>
<td>-0.0000</td>
<td>0.0002</td>
</tr>
<tr>
<td>UNIT 13</td>
<td>-0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 14</td>
<td>0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>UNIT 15</td>
<td>-0.0000</td>
<td>0.0001</td>
</tr>
<tr>
<td>UNIT 16</td>
<td>-0.0000</td>
<td>0.0001</td>
</tr>
<tr>
<td>UNIT 17</td>
<td>0.0000</td>
<td>0.0004</td>
</tr>
<tr>
<td>UNIT 18</td>
<td>0.0000</td>
<td>-0.0001</td>
</tr>
<tr>
<td>UNIT 19</td>
<td>-0.0000</td>
<td>0.3758</td>
</tr>
<tr>
<td>UNIT 20</td>
<td>0.2398</td>
<td>-42.0407</td>
</tr>
</tbody>
</table>

### BIAS

<table>
<thead>
<tr>
<th>BIAS</th>
<th>0.5212</th>
<th>-0.0216</th>
<th>0.2434</th>
<th>1.1899</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>BIAS</th>
<th>-0.8530</th>
<th>6.9963</th>
</tr>
</thead>
</table>