Linear and Nonlinear Associative Memories for Parameter Estimation*

R. KALABA

Departments of Electrical and Biomedical Engineering, University of Southern California, Los Angeles, California 90089

Z. LICHTENSTEIN

and

T. SIMCHONY

Signal and Image Processing Institute, University of Southern California, Los Angeles, California 90089

L. TESFATSION

Departments of Economics and Mathematics, Iowa State University, Ames, Iowa 50011

Communicated by Robert Kalaba

ABSTRACT

This paper proposes the use of associative memories for obtaining preliminary parameter estimates for nonlinear systems. For each parameter vector $\mathbf{r}_i$ in a selected training set, the system equations are used to determine a vector $\mathbf{s}_i$ of system outputs. An associative memory matrix $\hat{\mathbf{M}}$ is then constructed which optimally, in the least squares sense, associates each system output vector $\mathbf{s}_i$ with its corresponding parameter vector $\mathbf{r}_i$. Given any observed system output vector $\mathbf{s}^*$, an estimate $\hat{\mathbf{r}}$ for the system parameters is obtained by setting $\hat{\mathbf{r}} = \hat{\mathbf{M}}\mathbf{s}^*$. Numerical experiments are reported which indicate the effectiveness of this approach, especially for the nonlinear associative memory case in which the training vectors $\mathbf{s}_i$ include not only the system output levels but also products of these levels. Training with noisy output vectors is shown to improve the accuracy of the parameter estimates when the

*The first author is partially supported by NIH Grant DK 33729, and the second and third authors are partially supported by NSF Grant MIP-84-51010 and matching funds from IBM, AT&T, and Hughes Aircraft Company; the third author is also supported by ECI Telecom Israel. Please address correspondence to Professor Leigh Tesfatsion, Department of Economics, Iowa State University, Ames, Iowa 50011-1070.

655 Avenue of the Americas, New York, NY 10010 0020-0255/92/$5.00

observation vectors $\mathbf{s}^*$ are noisy. If experimental data are available for use as the training set, the estimation procedure can be carried out without knowing the system equations.

1. INTRODUCTION

Parameter estimation problems for nonlinear systems are typically formulated as nonlinear optimization problems—for example, nonlinear least squares. Many of the batch procedures used to solve such problems are based on Newton's method, which requires good preliminary parameter estimates to guarantee convergence to the correct solution. Alternatively, recursive procedures such as extended Kalman filtering can be used; but again it is necessary to have good preliminary parameter estimates to ensure convergence in a reasonable amount of time.

In this paper we propose the use of associative memories [1–4] for obtaining preliminary parameter estimates for nonlinear systems characterized by a finite number of parameters. An information processor acts as an associative memory if, having stored the vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_q$ at the respective "addresses" $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_q$, it retrieves a response vector $\mathbf{r}$ which is close in some sense (L_2 norm for example) to $\mathbf{r}_i$ when it is stimulated with a vector $\mathbf{s}$ which is close to $\mathbf{s}_i$. We propose the construction of an associative memory matrix which associates each parameter vector in a selected training set with a corresponding vector of system outputs generated from the system equations. Actual system observations are then used as a stimulus to the associative memory matrix in order to obtain a response (estimate) for the actual parameters of the system.

How does the associative memory approach differ from the approach of nonlinear least squares (NLS)? In the latter approach it is supposed that a nonlinear relation holds between an $m \times 1$ output vector $\mathbf{s}$ and an $n \times 1$ parameter vector $\mathbf{r}$ of the form

$$\mathbf{s} = F(\mathbf{r}). \tag{1}$$

When an actual observation vector $\mathbf{s}^*$ is obtained, $\mathbf{r}$ is estimated by

$$\mathbf{r}^{\mathrm{NLS}} = \operatorname{argmin}_{\mathbf{r}} \|\mathbf{s}^* - F(\mathbf{r})\|^2, \tag{2}$$

where $\|\cdot\|$ denotes the L_2 norm. Note that $\mathbf{r}^{\mathrm{NLS}}$ is a function of both $\mathbf{s}^*$ and F:

$$\mathbf{r}^{\mathrm{NLS}} = \mathbf{r}^{\mathrm{NLS}}(\mathbf{s}^*, F). \tag{3}$$

For each new observation vector $\mathbf{s}^*$, a new minimization problem (2) must be solved.

The associative memory approach to parameter estimation proceeds quite differently. Before any actual observation vector is obtained, finitely many training cases are constructed. Specifically, for each parameter vector $\mathbf{r}_i$ in some selected finite set $\{\mathbf{r}_i | i = 1, \ldots, q\}$, the system function $F(\cdot)$ is used to generate a corresponding vector $\mathbf{s}_i = F(\mathbf{r}_i)$ of system outputs. These training vector pairs are used to construct "stimulus" and "response" matrices

$$\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_q)_{m \times q}, \qquad \mathbf{R} = (\mathbf{r}_1, \ldots, \mathbf{r}_q)_{n \times q}. \tag{4}$$

An "associative memory matrix" $\hat{\mathbf{M}}$ is then constructed which associates the training stimulus (output) vectors $\mathbf{s}_1, \ldots, \mathbf{s}_q$ with the corresponding training response (parameter) vectors $\mathbf{r}_1, \ldots, \mathbf{r}_q$, in the sense that $\hat{\mathbf{M}}\mathbf{S}$ is "close" to $\mathbf{R}$. For example, one might try[1]

$$\hat{\mathbf{M}} = \mathrm{argmin}_M \|\mathbf{MS} - \mathbf{R}\|^2, \tag{5}$$

where

$$\hat{\mathbf{M}} = \hat{\mathbf{M}}((\mathbf{r}_1, \mathbf{s}_1), \ldots, (\mathbf{r}_q, \mathbf{s}_q)). \tag{6}$$

When an actual observation vector $\mathbf{s}^*$ is obtained, the corresponding parameter vector $\mathbf{r}$ is estimated by

$$\mathbf{r}^{\mathrm{AM}} = \hat{\mathbf{M}}\mathbf{s}^*, \tag{7}$$

where

$$\mathbf{r}^{\mathrm{AM}} = \mathbf{r}^{\mathrm{AM}}(\mathbf{s}^*, \hat{\mathbf{M}}). \tag{8}$$

Note that $\hat{\mathbf{M}}$ depends on the training cases, but $\hat{\mathbf{M}}$ does not directly depend on the system function $F(\cdot)$. Thus, it is not essential to know $F(\cdot)$ in order to determine $\mathbf{r}^{\mathrm{AM}}$ in (8). One could, for example, use training cases obtained directly from experimental results. Also, as long as the training cases used to construct the associative memory matrix $\hat{\mathbf{M}}$ remain relevant, new parameter

[1]Other characterizations for $\hat{\mathbf{M}}$ are possible, and sometimes even essential. See Section 4, below.

estimates $\mathbf{r}^{AM}$ can be successively generated for new observation vectors $\mathbf{s}^*$ by repeated applications of the simple matrix operation (7).

NLS and associative memory thus represent two distinct approaches to the problem of obtaining parameter estimates for nonlinear systems. On the face of it, however, it seems difficult to believe that reasonable parameter estimates can be obtained for nonlinear systems by means of a simple linear operation such as (7).

The crucial point is that the associative memory matrix $\hat{\mathbf{M}}$ encodes the nonlinear inverse mapping between the outcome $\mathbf{s}_i = F(\mathbf{r}_i)$ and the parameter vector $\mathbf{r}_i$ for each parameter vector $\mathbf{r}_i$ in the selected training grid $\mathbf{r}_1,\ldots,\mathbf{r}_q$. The training procedure can be compared with the maximum likelihood method for associating the most "likely" parameter vector with any given outcome.

As established in [3], an associative memory matrix $\hat{\mathbf{M}}$ can in principle be determined so that $\hat{\mathbf{M}}\mathbf{s}_i$ is arbitrarily close to $\mathbf{r}_i$ for each stimulus-response pair $(\mathbf{s}_i,\mathbf{r}_i)$, $i=1,\ldots,q$, assuming that the stimulus vectors $\mathbf{s}_i$ have been suitably "preprocessed" (see Section 4, below). If the associative memory matrix $\hat{\mathbf{M}}$ also has good interpolation properties, then any stimulus vector $\mathbf{s}^*$ which is a weighted combination of the training stimulus vectors $\mathbf{s}_i$ should be mapped by $\hat{\mathbf{M}}$ into a weighted combination $\mathbf{r}^{AM}$ of the training response vectors $\mathbf{r}_i$ which is close to the actual response vector $\mathbf{r}^*$. In this case, the estimate $\mathbf{r}^{AM}$ should also provide a good preliminary estimate for solving the NLS minimization problem (2) by means of some successive approximation scheme.

Much additional theoretical work is needed to turn these heuristic statements into rigorous arguments. This paper provides an important prerequisite: nontrivial numerical experiments demonstrating that the associative memory approach is indeed capable of generating parameter estimates for nonlinear systems which are surprisingly accurate.

The particular problem we study—a problem in image processing—is set out in Section 2. For simplicity, we first consider the case in which the training stimulus vectors $\mathbf{s}_i$ are not preprocessed (i.e., the associative memory is "linear") and the system observations are not corrupted with noise.

Numerical experiments for this case are reported in Section 3. These experiments were originally undertaken with low expectations; we anticipated moving quickly to a more sophisticated artificial neural network approach as described, for example, in Refs. [5–7]. We were thus astonished to find that the linear associative memory approach yielded reasonably accurate parameter estimates for our nonlinear image processing problem, even with a small number of observation times.

Subsequent experimentation revealed the possibility and desirability of modifying the basic linear associative memory approach in order to handle noisy observations and more extensive learning intervals for the parameters.

The treatment of noisy observations is taken up in Section 4. Other researchers [8–9] have suggested the use of mean-square error and/or singular value decomposition methods for handling noise in associative memory problems. We found, however, that the linear associative memory approach yielded reasonably accurate parameter estimates despite noisy observations whenever the training was carried out in accordance with the following simple maxim: *Construct the associative memory matrix using training stimulus vectors corrupted by the same kind of noise that one anticipates will corrupt the actual system observations.*

In the course of these experiments with linear associative memory matrices, we saw that the accuracy of the parameter estimates tended to decrease when a larger grid of training parameter vectors was used, even when the coarseness of the grid was not increased. Section 5 reports on various experiments undertaken using a modified "nonlinear" associative memory approach which is potentially more suitable for handling larger training parameter grids.

Specifically, following Poggio [2], we constructed various kth-order polynomial associative memory matrices using processed training stimulus vectors whose components included not only system output levels but also up through kth-order products of these levels. Using a second-order polynomial associative memory matrix in place of a linear associative memory matrix, we found a substantial improvement in the resulting parameter estimates for both noisy and noise-free observations. Indeed, the estimates were so accurate that further refinement (e.g., through an NLS procedure) seemed unnecessary. The use of a third-order polynomial associative memory matrix yielded only small additional improvements for our problem.

Concluding comments are given in Section 6.

2. AN ILLUSTRATIVE PROBLEM FROM IMAGE PROCESSING

Suppose a point is rotating in two-dimensional space with angular velocity ω and initial phase ϕ. Starting from a given position (x_0, y_0) at time $t = 0$, the position of the point evolves in accordance with the system equations

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} \cos(\omega t + \phi) & -\sin(\omega t + \phi) \\ \sin(\omega t + \phi) & \cos(\omega t + \phi) \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}. \tag{9}$$

At each time $t = 1, \ldots, p$, a noisy observation $(x^*(t), y^*(t))$ is obtained on the position of the point in accordance with the measurement equations

$$\begin{bmatrix} x^*(t) \\ y^*(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} + \begin{bmatrix} n_1(t) \\ n_2(t) \end{bmatrix}, \tag{10}$$

where $n_1(t)$ and $n_2(t)$ are observation noise. Given the initial position of the point at time 0 and the $2p$-dimensional noisy observation vector

$$\mathbf{s}^* = \begin{pmatrix} x^*(1) \\ y^*(1) \\ \vdots \\ x^*(p) \\ y^*(p) \end{pmatrix}, \tag{11}$$

the problem is to estimate the angular velocity ω and the initial phase ϕ of the rotating point.

A classical approach to this parameter estimation problem is to pose it as a nonlinear least squares problem in which the expression

$$\sum_{t=1}^{p} \{[x^*(t) - x_0 \cdot \cos(\omega t + \phi) + y_0 \cdot \sin(\omega t + \phi)]^2$$
$$+ [y^*(t) - x_0 \cdot \sin(\omega t + \phi) - y_0 \cdot \cos(\omega t + \phi)]^2\} \tag{12}$$

is to be minimized with respect to ω and ϕ. To solve this problem, some form of Newton's method [10] or extended Kalman filtering is often used. A major drawback of these approaches is the need to have good preliminary estimates for the true parameters ω and ϕ.

We will now outline a linear associative memory procedure for obtaining preliminary estimates for ω and ϕ under the assumption that the noise components in (10) are zero; noise effects are taken up in Section 4. The basic idea is to find an inverse mapping from the observations to the parameters using a supervised learning procedure.

The first step in the procedure is the construction of a finite set of training vectors

$$\mathbf{r}_i = \begin{pmatrix} \omega \\ \phi \end{pmatrix}_i, \qquad \mathbf{s}_i = \begin{pmatrix} x(1) \\ y(1) \\ \vdots \\ x(p) \\ y(p) \end{pmatrix}_i, \qquad i = 1, 2, \ldots, q, \tag{13}$$

where $\mathbf{r}_i$ is a potential vector of system parameters, $\mathbf{s}_i$ is the corresponding system output generated from the system equations (9), and $\mathbf{q}$ is the number of training cases. The training vectors (13) are used to form the response and stimulus matrices

$$\mathbf{R} = (\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_q)_{2\times q}, \qquad \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_q)_{2p\times q}. \tag{14}$$

Given the response and stimulus matrices (14), the optimal (least-squares) linear mapping between these two matrices is determined by solving

$$\min_M \|\mathbf{MS} - \mathbf{R}\|^2, \tag{15}$$

where $\|\cdot\|$ denotes the L_2 norm. As shown in [11], the minimum-norm solution to problem (15) is

$$\hat{\mathbf{M}} = \mathbf{RS}^+, \tag{16}$$

where $\mathbf{S}^+$ is the Moore–Penrose generalized inverse of $\mathbf{S}$. Given any observation vector $\mathbf{s}^*$ for system (9), an estimate $\hat{\mathbf{r}}$ for the parameter vector $\mathbf{r} = (\omega, \phi)^T$ is determined by

$$\hat{\mathbf{r}} = \hat{\mathbf{M}}\mathbf{s}^*. \tag{17}$$

3. LINEAR ASSOCIATIVE MEMORY WITH NOISE-FREE OBSERVATIONS

This section reports on several numerical experiments using a linear associative memory approach to obtain parameter estimates for system (9) with noise-free observations.

For each different experiment, basic "learning intervals" $(\omega_{\min}, \omega_{\max})$ and $(\phi_{\min}, \phi_{\max})$ were first specified for the parameters ω and ϕ. These learning intervals were subdivided using fixed step sizes $\Delta\omega$ and $\Delta\phi$, so that the learning rectangle formed by the product of the learning intervals was covered by a grid of points of the form $(\omega_{\min} + j\,\Delta\omega, \phi_{\min} + k\,\Delta\phi)$. These grid points were then used as the training parameter vectors $\mathbf{r}_1, \ldots, \mathbf{r}_q$ for the experiment.

For example, given the learning interval $(-1,+1)$ for ω with step size $\Delta\omega = 0.2$ and the learning interval $(-0.5,0.5)$ for ϕ with step size $\Delta\phi = 0.1$, the training grid consists of 121 parameter vectors $\mathbf{r}_i = (\omega_i, \phi_i)^T$.

For each training parameter vector $\mathbf{r}_i$ in the training grid, the system equations (9) were used to generate a corresponding vector $\mathbf{s}_i$ of system outputs. Response and stimulus matrices were then constructed as in (14), and the associative memory matrix $\hat{\mathbf{M}}$ was determined as in (16) using these noise-free training cases.[2]

Next, about 400 "test parameter vectors" $\mathbf{r} = (\omega, \phi)^T$ were evenly interspersed among the training parameter vectors constituting the training grid. For each test parameter vector $\mathbf{r}$ in this test grid, a noise-free $2p$-dimensional observation vector

$$\mathbf{s} = \begin{pmatrix} x(1) \\ y(1) \\ \vdots \\ x(p) \\ y(p) \end{pmatrix} \tag{18}$$

was generated using the system equations (9). A parameter estimate $\hat{\mathbf{r}} = (\hat{\omega}, \hat{\phi})^T$ was then determined as $\hat{\mathbf{r}} = \hat{\mathbf{M}}\mathbf{s}$ in accordance with (17). The *relative estimate error* for ω was defined to be

$$\delta_\omega \triangleq \frac{|\omega - \hat{\omega}|}{\omega_{\max} - \omega_{\min}}. \tag{19}$$

An analogous expression was used to define the relative estimate error for ϕ. Finally, the maximum relative estimate errors for both ω and ϕ were calculated over all of the test parameter vectors $\mathbf{r}$.

Table 1 reports on some of these experiments. For each of the three experiments presented in Table 1, the initial position of the point was specified to be $(x_0, y_0) = (1,1)$ and the learning intervals were specified to be $(\omega_{\min}, \omega_{\max}) = (-1,1)$ and $(\phi_{\min}, \phi_{\max}) = (-0.5,0.5)$. All computations, including the computation of the generalized inverse, were done using PRO-MATLAB.

[2]An example of one such experimentally obtained matrix, together with parameter estimation results obtained with the matrix, can be found in the Appendix to this paper.

TABLE 1

Linear Associative Memory Results with Noise-Free Observations

Expt.	$\Delta\omega$	$\Delta\phi$	Observation length p	No. of training cases	Maximum δ_ω	Maximum δ_ϕ
#1	0.2	0.1	5	121	0.092	0.053
#2	0.1	0.1	5	231	0.080	0.046
#3	0.1	0.1	10	231	0.040	0.007

As indicated in Table 1, reasonably good parameter estimates were obtained for system (9) using a linear associative memory matrix with five observation times. The accuracy of the parameter estimates improved considerably when the observation length p was increased from five to ten observation times.

The accuracy of the parameter estimates tended to be highest for test parameter vectors located near the center of the training parameter grid and lowest for test parameter vectors located near the boundaries of the training parameter grid. Taking parameter vectors successively further outside of the convex hull of the training parameter grid resulted in a rather rapid degradation in the resulting parameter estimates; such points were not included in the test parameter grid for Table 1.

The linear associative memory matrix thus appears to perform fairly well as an interpolator over the training parameter grid, but not as an extrapolator outside of this grid. Additional numerical results illustrating these points for experiment #3 in Table 1 can be found in the Appendix.

4. LINEAR ASSOCIATIVE MEMORY WITH NOISY OBSERVATIONS

Usually the observation vectors $\mathbf{s}^*$ for system (9) will be corrupted by nonzero noise terms. This raises several questions. First, how does observation noise affect the accuracy of the parameter estimates obtained using a linear associative memory matrix constructed from noise-free training vectors?

As might be expected, the parameter estimates obtained for noisy observation vectors using noise-free training vectors can be quite poor. For example, in one series of experiments the noise components $n_1(t), n_2(t)$ for the system (9) observation vectors $\mathbf{s}^*$ were generated using a pseudorandom number generator for a Gaussian distribution with zero mean and with standard deviation equal to 0.1. The estimates $\hat{\mathbf{r}} = (\hat{\omega}, \hat{\phi})^T$ that were obtained as in (17) using the resulting noisy observation vectors $\mathbf{s}^*$ but an associative memory matrix $\hat{\mathbf{M}}^0$ constructed from noise-free training cases were highly inaccurate. For example, using Monte-Carlo analysis, the estimated standard deviation of the relative estimate error δ_ω for ω over the learning interval $(-1, 1)$ was 16.6.

A similar result was obtained for the estimated standard deviation of the relative estimate error δ_ϕ for ϕ.[3]

Second, given that noise is a problem, how might its effects be offset? One possibility is to introduce a minimum mean-square error criterion for $\mathbf{M}$ in place of the criterion (15). Alternatively, one could attempt to increase the numerical stability of the parameter estimates by using a singular value decomposition method to modify the calculation of the generalized inverse $\mathbf{S}^+$ in (16). These approaches are investigated in Refs. [8, 9] for noisy associative memory problems where the number q of stimulus-response training pairs is less than the dimension of the stimulus vectors and the only objective is to achieve good encoding and recall of these q stimulus-response associations.

It seemed to us, however, that a simpler way to deal with noisy observations might be possible for associative memory problems in which the encoding and recall of training cases is viewed as a means to a further end—achieving good interpolative capability—and consequently the number of training cases is not restricted to be less than the dimension of the stimulus vectors. Intuitively, if an information processor is going to be usefully trained to associate new stimuli with appropriate responses, then presumably the training stimuli should resemble the stimuli which will actually be encountered in practice. For the particular image processing problem at hand, this translates into the following maxim: *Construct the associative memory matrix using training stimulus vectors corrupted by the same kind of noise that one anticipates will corrupt the actual system observations.*

Surprisingly, we found that the use of noisy training vectors substantially reduced the errors in our parameter estimates. Specifically, for each parameter (response) vector $\mathbf{r}_i$ in the training set, we corrupted the corresponding output (stimulus) vector $\mathbf{s}_i$ with a noise vector whose components were generated using a pseudorandom number generator for a Gaussian distribution with zero mean and with standard deviation equal to 0.1. A stimulus

[3] More precisely, the Monte Carlo analysis consisted of the following steps. For each test parameter vector $\mathbf{r}$, a system output vector $\mathbf{s}$ was calculated using the system equations (9). Using this one system output vector $\mathbf{s}$, 100 noisy system observation vectors $\mathbf{s}^* = \mathbf{s} + \mathbf{n}$ were then created as in (10) and (11), where the i.i.d. components of the noise vectors $\mathbf{n}$ were generated using a Gaussian pseudorandom number generator with mean zero and standard deviation equal to 0.1. For each of these observation vectors $\mathbf{s}^*$, an estimated parameter vector $\hat{\mathbf{r}}$ was determined as in (17), using the associative memory matrix $\hat{\mathbf{M}}^0$ generated from noise-free training vectors; and the corresponding relative estimate errors for ω and ϕ were calculated. The mean and standard deviation of these relative estimate errors over all test parameter vectors, with 100 samples for each test parameter vector, were then determined. [As will be seen below in Equation (22), an analytical expression is available for the standard deviation.]

matrix $\mathbf{S}^{n*}$ was then constructed using these noisy training stimulus vectors. Finally, an associative memory matrix $\hat{\mathbf{M}}^{n*}$ was determined as in (15) using this noisy stimulus matrix. The estimates $\hat{\mathbf{r}}$ that were subsequently obtained for noisy observation vectors $\mathbf{s}^*$ corrupted by i.i.d. Gaussian noise, again with zero mean and with standard deviation equal to 0.1, were now reasonably precise. For example, the estimated standard deviation of the relative estimate error for ω over the learning interval $(-1,1)$ was 0.0367.

To provide motivation for these latter results, consider the effects of observation noise on the parameter estimates obtained via an associative memory matrix. Given a noisy observation vector $\mathbf{s}^* = \mathbf{s}+\mathbf{n}$, where $\mathbf{s}$ and $\mathbf{n}$ are the noise-free and noisy parts of $\mathbf{s}^*$, the parameter estimate generated as in (15) is

$$\hat{\mathbf{r}} = (\hat{\mathbf{r}}^s + \hat{\mathbf{r}}^n) = \hat{\mathbf{M}}(\mathbf{s}+\mathbf{n}), \tag{20}$$

where $\hat{\mathbf{r}}^s$ and $\hat{\mathbf{r}}^n$ are the noise-free and noisy parts of $\hat{\mathbf{r}}$.

If the components of the associative memory matrix $\hat{\mathbf{M}}$ are large, it is obvious from (20) that the resulting parameter estimates $\hat{\mathbf{r}}$ will be highly sensitive to observation noise, whatever its source. In other words, even a small perturbation $\mathbf{n}$ in the observation vector $\mathbf{s}^*$ might result in a large change $\hat{\mathbf{r}}^n = \hat{\mathbf{M}}\mathbf{n}$ in the parameter estimate $\hat{\mathbf{r}}$—a highly undesirable situation. Consequently, once the possibility of imperfect measurement is recognized, keeping the magnitudes of the components of the associative memory matrix small becomes an important criterion in addition to the basic criterion of obtaining good training case associations.[4]

In what sense, if any, does the use of noisy training stimulus vectors achieve a balance between these two potentially conflicting criteria?

Let $\mathbf{S}^n = \mathbf{S}+\mathbf{N}$ denote a $2p \times q$ stimulus matrix which is the sum of a given matrix $\mathbf{S}$ of noise-free training stimulus vectors and a given matrix $\mathbf{N}$ of noise components. Also, let $\mathbf{R}$ denote a given $2 \times q$ matrix of training response vectors. The objective function for the choice of the associative memory matrix $\mathbf{M}$ can then be written in the expanded form

$$\begin{aligned}\|\mathbf{MS}^n - \mathbf{R}\|^2 &\equiv \mathrm{Trace}\big([\mathbf{MS}^n - \mathbf{R}][\mathbf{MS}^n - \mathbf{R}]^T\big)\\ &= \mathrm{Trace}\big([\mathbf{M}(\mathbf{S}+\mathbf{N}) - \mathbf{R}][\mathbf{M}(\mathbf{S}+\mathbf{N}) - \mathbf{R}]^T\big)\\ &= \|\mathbf{MS}-\mathbf{R}\|^2 + \mathrm{Trace}(\mathbf{MNN}^T\mathbf{M}^T) + 2\,\mathrm{Trace}\big(\mathbf{MN}[\mathbf{MS}-\mathbf{R}]^T\big).\end{aligned} \tag{21}$$

[4]This observation is of course just a special case of a long-recognized point in linear estimation theory—the desirability of reducing the norm of a linear estimator in order to enhance the numerical stability of the resulting estimates.

In each experiment we ran with noisy observation vectors $\mathbf{s}^*$, the associative memory matrix $\hat{\mathbf{M}}^n$ which minimized the objective function (21) had smaller component entries (and subsequently yielded more precise parameter estimates) than the corresponding matrix $\hat{\mathbf{M}}$ which minimized a modified noise-free form of the objective function (21) with $\mathbf{N}$ replaced by a matrix of zeros. The intuitive reason for these findings is suggested by a consideration of the average form of the objective function (21) across experiments.

The components n_{jk} of the noise matrix $\mathbf{N}$ in (21) are given numbers representing realized perturbations in the system observations for the particular experiment under consideration. No assumptions are made concerning the source of these perturbations. Suppose instead that (21) is interpreted as an *ex ante* objective function in which the components of $\mathbf{N}$ are yet-to-be realized i.i.d. random variables with mean zero and variance σ^2. Suppose also that the objective is to choose an associative memory matrix $\mathbf{M}$ to minimize the *expected value* of the objective function (21). In other words, suppose that a mean-square error objective function is to be used to select $\mathbf{M}$.

To determine the form which this mean-square error objective function takes, consider first a preliminary technical observation. Let $\mathbf{U} \equiv \mathbf{MN}$, where $\mathbf{M}$ is any given $2 \times 2p$ matrix, and let u_{ik} denote the ith component of the kth column of $\mathbf{U}$. Then u_{ik} has zero mean and a variance σ_{ik}^2 given by

$$\sigma_{ik}^2 = \mathbf{E}\left\{ \sum_j (M_{ij} \cdot n_{jk})^2 \right\} = \sigma^2 \sum_j M_{ij}^2. \tag{22}$$

The variance of u_{ik} is therefore proportional to the sum of the squares of the entries of $\mathbf{M}$ in the ith row, and this variance does not depend on k.

Using (22), it can be shown that the mean-square error objective function takes the form

$$\begin{aligned} &E\{\mathrm{Trace}([\mathbf{M}(\mathbf{S}+\mathbf{N})-\mathbf{R}][\mathbf{M}(\mathbf{S}+\mathbf{N})-\mathbf{R}]^T)\} \\ &= \|\mathbf{MS}-\mathbf{R}\|^2 + q \cdot \sigma^2 \sum_{i,j} M_{ij}^2. \end{aligned} \tag{23}$$

Note that the sum of the squares of the components of $\mathbf{M}$, multiplied by the variance σ^2 of the noise components and the number of training cases q, enters additively into the right-hand expression for the mean-square error objective function. Consequently, this objective function is a penalty function which takes into account two different criteria for the choice of $\mathbf{M}$: Namely, achieve good training case associations (i.e., choose $\mathbf{M}$ so that $\mathbf{MS} \simeq \mathbf{R}$), and keep the magnitudes of the components of $\mathbf{M}$ small. One would therefore expect the components of the matrix $\mathbf{M}$ which minimizes (23) for the noisy case $\sigma^2 > 0$ to be reduced in magnitude in comparison with the noise-free case $\sigma^2 = 0$.

The mean-square error objective function (23) is the expected value of the objective function (21) actually used to determine the associative memory matrix $\mathbf{M}$ on an experiment-by-experiment basis. Consequently, one would expect to see on average—i.e., over the ensemble of all experiments—that the components of the associative memory matrices $\hat{\mathbf{M}}^n$ which minimize (21) have smaller magnitudes than the components of the associative memory matrices $\hat{\mathbf{M}}$ which minimize the noise-free version of (21).

In fact, this anticipation has been borne out in each of our experiments, not just on average.[5] For example, the sum of squares of the components of the two associative memory matrices compared earlier in this section were 1.6514 for $\hat{\mathbf{M}}^{n*}$ learned with noise and 1.3168×10^5 for $\hat{\mathbf{M}}^0$ learned without noise. An explicit component-by-component comparison of a noisy matrix $\hat{\mathbf{M}}^n$ with its corresponding noise-free matrix $\hat{\mathbf{M}}$ is provided in the Appendix.

5. POLYNOMIAL ASSOCIATIVE MEMORIES

We saw in Sections 3 and 4 that linear associative memory matrices yield reasonably good parameter estimates for system (9) when learning intervals are not too extensive and when noisy training vectors are used in anticipation of noisy observation vectors. Furthermore, the accuracy of the parameter estimates increases as the observation length increases.

However, the linear associative memory approach does not work well in all cases. For example, the accuracy of the parameter estimates obtained with linear associative memory matrices rapidly drops off when the test parameter vectors are taken outside of the training grid. Moreover, as indicated in Table 2, below, increasing the learning intervals significantly reduces the accuracy of the parameter estimates. For all of the experiments reported in Table 2, the observation length p is equal to 10. As before, test parameter vectors $\mathbf{r}$ were evenly interspersed among the training parameter vectors $\hat{\mathbf{r}}$ constituting the training grid.

The results reported in Table 2 suggest that the specification of the training parameter grid must be done with care if good parameter estimates are to be obtained using a linear associative memory matrix. For example, when the

[5]A possible explanation for this strong finding is that our method of generating the noisy training stimulus vectors mimicked a sampling procedure from a stationary distribution. In this case it may be possible to establish that the criterion function (21) provides a more direct approximation to a mean-square error criterion function.

TABLE 2

Linear Associative Memory Results with Increased Learning Intervals

Expt.	$(\omega_{\min}, \omega_{\max})$	$\Delta\omega$	$(\phi_{\min}, \phi_{\max})$	$\Delta\phi$	No. of training cases	Maximum δ_ω	Maximum δ_ϕ
#1	(−1,1)	0.2	(−0.5,0.5)	0.2	66	0.040	0.010
#2	(−1,1)	0.2	(−1,1)	0.2	121	0.180	0.027
#3	(−2,2)	0.2	(−1,1)	0.2	231	0.375	0.219
#4	(−2,2)	0.1	(−1,1)	0.1	861	0.356	0.188
#5	(−2,2)	.05	(−1,1)	.05	3,321	0.352	0.240

learning intervals were increased in going from experiment #1 to experiment #3, the precision of the parameter estimates was substantially reduced; and this marked degradation persisted even when finer mesh sizes were used for the training parameter grid, as in experiments #4 and #5.

If large learning intervals are required, one might consider reverting to a more sophisticated artificial neural network approach—for example, a Rumelhart net [5,6] in which internal layers of hidden units with threshholding operations are used to provide a representation for the possibly nonlinear relation connecting input (stimulus) and output (response) vectors. One difficulty with the latter approach is that it is not possible to know in advance just how many hidden units and layers will suffice for the problem at hand. Moreover, obtaining the "connective strengths" for a Rumelhart net is a nonlinear optimization problem for which convergence is not always assured.

Poggio [2] provides an interesting alternative way to proceed which constitutes a middle-ground between Kohonen's linear associative memory approach and the more computationally demanding Rumelhart approach. Poggio processes the training stimulus vectors prior to their use in constructing an associative memory matrix in a way which magnifies the dissimilarities among these vectors. Using Poggio's approach, it may be possible to produce an associate memory matrix which generates reasonably accurate parameter estimates even when the training parameter grid is quite large.

Specifically, Poggio suggests that the associative mapping from the observations to the parameters be approximated by a recursively generated kth-order polynomial. The crucial step in the Poggio method is the initial transformation of each stimulus vector into a processed vector that includes up through kth-order distinct products of the components of the stimulus vector. For example, if the stimulus vector **s** consists of the three scalar components s_1, s_2, and s_3, then the second-order processed vector for **s** takes the form

$$\mathbf{z} = (s_1, s_2, s_3, s_1 \cdot s_1, s_1 \cdot s_2, s_1 \cdot s_3, s_2 \cdot s_2, s_2 \cdot s_3, s_3 \cdot s_3)^T. \tag{24}$$

TABLE 3

Polynomial Associative Memory Results with Noise-Free Observations

Expt.	Order k	No. of elements in z_i	No. of training cases	Maximum δ_ω	Maximum δ_ϕ
#1	1	10	187	0.4220	0.3980
#2	2	65	187	0.0019	0.0727
#3	3	285	187	0.0005	0.0450

To apply the Poggio method to the problem of obtaining preliminary parameter estimates for system (9), we first construct a set of kth-order processed training stimulus vectors

$$\mathbf{z}_i \triangleq \begin{pmatrix} \mathbf{s}_i^1 \\ \mathbf{s}_i^2 \\ \vdots \\ \mathbf{s}_i^k \end{pmatrix}, \qquad i = 1, \ldots, q, \tag{25}$$

where $\mathbf{s}_i^j$ includes all of the distinct jth-order products of the components of the stimulus vector $\mathbf{s}_i^1 \equiv \mathbf{s}_i$ taken in an agreed-upon order. In order to determine a kth-order polynomial associative memory matrix $\hat{\mathbf{M}}$, a minimization problem of the form (15) is again solved; but now the columns of the stimulus matrix consist of the processed training stimulus vectors $\{\mathbf{z}_i,\ i = 1, \ldots, q\}$ in place of the unprocessed training stimulus vectors $\{\mathbf{s}_i,\ i = 1, \ldots, q\}$.

Table 3 presents some results for problem (9) using the learning intervals $\omega \in (-2, 2)$ and $\phi \in (-1, 1)$, the grid mesh specifications $\Delta\omega = 0.25$ and $\Delta\phi = 0.2$, the observation length $p = 5$, and polynomial associative memory matrices of first, second, and third order. As in previous experiments, test parameter vectors were interspersed throughout the grid of training parameter vectors.

As indicated in Table 3, the increase in the accuracy of the parameter estimates using a second-order polynomial associative memory matrix in place of a linear associative memory matrix was substantial. Indeed, the resulting parameter estimates were so accurate that the qualifier "preliminary" seemed unnecessary. The use of a third-order polynomial associative memory matrix contributed only a small additional improvement. The findings reported in Table 3 appear to be robust for problem (9); similar results were obtained in one experiment after another.

Another issue still has to be considered: How should potential noise in the observations be handled when polynomial associative memory matrices are to be used?

Experimentally, we found that good parameter estimates were obtained by adhering to the simple Section 4 maxim: *Use noisy training stimulus vectors when noise is anticipated in the observations*. To illustrate our findings, parameter estimates obtained by use of two different second-order polynomial associative memory matrices will now be compared. The first matrix is an associative memory matrix $\hat{\mathbf{M}}$ constructed from *noise-free* processed training stimulus vectors $\mathbf{z}_i$, and the second matrix is an associative memory matrix $\hat{\mathbf{M}}^n$ constructed from *noisy* processed training stimulus vectors $\mathbf{z}_i^*$.

Specifically, we compare $\hat{\mathbf{M}}$ against $\hat{\mathbf{M}}^n$ for experiment #2 in Table 3. Other than the use of noisy training vectors for the construction of $\hat{\mathbf{M}}^n$, all specifications for the learning procedure were the same for both matrices. In particular, the test grid in each case consisted of about 400 test parameter vectors $\mathbf{r} = (\omega, \phi)^T$ interspersed among the training parameter vectors constituting the training grid.[6]

When the system observations were noise-free, the parameter estimates obtained by means of the noise-free associative memory matrix $\hat{\mathbf{M}}$ were somewhat better than the parameter estimates obtained by means of the noisy associative memory matrix $\hat{\mathbf{M}}^n$. Specifically, the maximum relative estimate errors for the parameter estimates obtained using $\hat{\mathbf{M}}$ were 0.002 for δ_ω and 0.073 for δ_ϕ. Using $\hat{\mathbf{M}}^n$, these maximum errors increased to 0.037 for δ_ω and to 0.234 for δ_ϕ.

We next report on the performance of the two matrices when the system observations were noisy. Using Monte Carlo analysis, we computed the means and standard deviations of the relative estimate errors δ_ω and δ_ϕ obtained for the test parameter grid, first using the noise-free matrix $\hat{\mathbf{M}}$, and then using the noisy matrix $\hat{\mathbf{M}}^n$. In each case we used second-order processed observation

[6] The noisy associative memory matrix $\hat{\mathbf{M}}^n$ was constructed as follows. For each training parameter vector $\mathbf{r}_i$ for experiment #2 in Table 3, the system equations (9) were used to generate a noise-free stimulus vector $\mathbf{s}_i$. As illustrated in (10) and (11), this stimulus vector was then corrupted with noise generated by means of a pseudorandom number generator for a Gaussian distribution with zero mean and with standard deviation equal to 0.1. The resulting noisy training vector $\mathbf{s}_i^*$ was then processed as in (25) for $k = 2$, yielding a noisy processed stimulus vector $\mathbf{z}_i^*$. The complete set of noisy processed stimulus vectors $\mathbf{z}_i^*$ was then used to construct a training stimulus matrix $\mathbf{Z}^n$. Finally, the associative memory matrix $\hat{\mathbf{M}}^n$ was determined in accordance with the objective function (15) with $\mathbf{Z}^n$ in place of $\mathbf{S}$.

vectors corrupted by i.i.d. zero-mean Gaussian noise with standard deviation equal to 0.1.[7]

Figure 1 reports on the results obtained for the test parameter vectors $\mathbf{r}=(\omega,\phi)^T$ constructed with $\phi=0.33$ and with ω ranging from -2.0 to $+2.0$. The means of the relative estimate errors obtained for $\hat{\mathbf{M}}$ and $\hat{\mathbf{M}}^n$ were similar. However, the standard deviations of the relative estimate errors were much larger for $\hat{\mathbf{M}}$ than for $\hat{\mathbf{M}}^n$. Recalling the discussion in Section 5, it is interesting to note that the sums of the squares of the matrix components were 29.17 for $\hat{\mathbf{M}}$ but only 4.96 for $\hat{\mathbf{M}}^n$.

These results for second-order polynomial associative memory matrices support our finding in Section 4 for linear associative memory matrices: Namely, the magnitudes of the components of the associative memory matrix are reduced, and the precision of the resulting parameter estimates is substantially improved, when the matrix is trained using stimulus vectors corrupted by the same kind of noise as corrupts the actual system observations. Nevertheless, it is not altogether clear *why* the use of noisy training vectors is so effective when processed stimulus vectors are used. The motivation provided for the Section 4 finding relied on the use of noisy (unprocessed) stimulus vectors $\mathbf{s}^*$ corrupted by additive i.i.d. noise terms $\mathbf{n}$. However, the processing operation considerably complicates the noise characteristics of the resulting processed stimulus vectors.

For example, suppose $\mathbf{z}^*=\mathbf{s}(\mathbf{z})+\mathbf{n}(\mathbf{z})$ is a noisy second-order processed stimulus vector generated as in (25) with $k=2$ from a noisy unprocessed stimulus vector $\mathbf{s}^*=\mathbf{s}+\mathbf{n}$. Since the components of $\mathbf{z}^*$ include products of the form $(s_j+n_j)\cdot(s_l+n_l)$, the components of the noise term $\mathbf{n}(\mathbf{z})$ for $\mathbf{z}^*$ include terms such as $s_j n_l + s_l n_j + n_j n_l$. Consequently, the components of $\mathbf{n}(\mathbf{z})$ are not mutually independent, and their mean need not be zero. Also, the covariance

[7]The means and standard deviations of the relative estimate errors for $\hat{\mathbf{M}}$ were calculated using the following steps. For each test parameter vector $\mathbf{r}=(\omega,\phi)^T$, the system equations (9) were used to generate a noise-free stimulus vector $\mathbf{s}$. This one noise-free stimulus vector was then used to construct 100 noisy stimulus vectors of the form $\mathbf{s}^*=\mathbf{s}+\mathbf{n}$, where the i.i.d. components of the noise vectors $\mathbf{n}$ were generated by means of a pseudorandom number generator for a Gaussian distribution with zero mean and with standard deviation equal to 0.1. These 100 noisy stimulus vectors $\mathbf{s}^*$ were then used to generate 100 second-order processed observation vectors $\mathbf{z}^*$ as in (25). For each of the 100 processed observation vectors $\mathbf{z}^*$, we generated a parameter estimate $\hat{\mathbf{r}}^*=\hat{\mathbf{M}}\mathbf{z}^*$. We then computed the mean and standard deviation of the relative estimate error δ_ω for the test parameter vector $\mathbf{r}$ based on this sample of 100 parameter estimates $\hat{\mathbf{r}}^*$, and similarly for the relative estimate error δ_ϕ. An analogous procedure was carried out to obtain the means and standard deviations of the relative estimate errors for $\hat{\mathbf{M}}^n$.

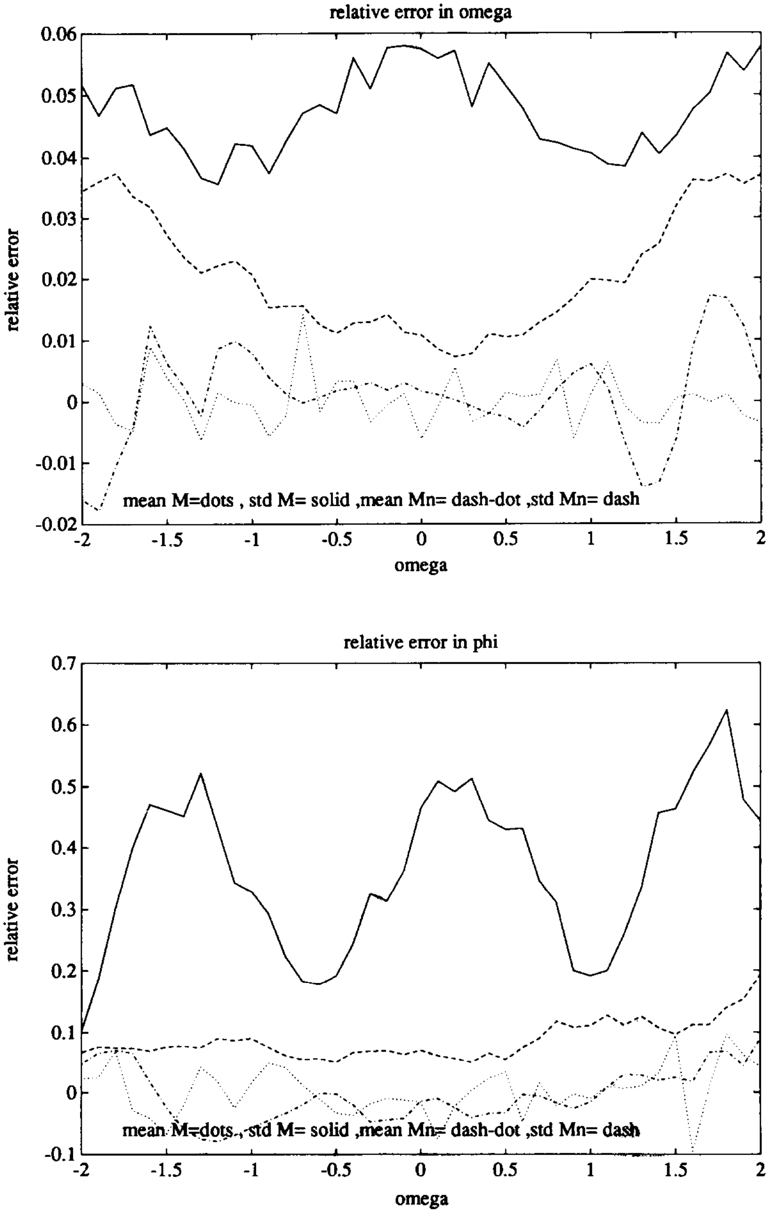

Fig. 1. Comparison of the relative estimate error means and standard deviations for the second-order polynomial associative matrix $\hat{\mathbf{M}}$ learned without noise and the second-order polynomial associative matrix $\hat{\mathbf{M}}^n$ learned with noise

matrix for $\mathbf{n}(\mathbf{z})$ depends on the basic noise-free observation vector $\mathbf{s}$ as well as on higher moments of the basic noise vector $\mathbf{n}$.

6. DISCUSSION

The numerical experiments reported here for a two-dimensional image processing problem suggest that linear and nonlinear associative memories provide a powerful new tool for obtaining preliminary parameter estimates for nonlinear systems. A linear associative memory approach [1] worked well when the training vectors were adroitly chosen and the noise in the observation vectors was not excessive. When the linear associative memory approach failed, the use of a second-order polynomial associative memory approach [2] yielded substantially improved parameter estimates. The latter estimates were in fact so accurate that recourse to additional refinements seemed unnecessary.

In all of the numerical experiments, noise in the observation vectors was effectively handled by the use of noisy training vectors in the learning stage. A more elaborate statistical and/or singular value decomposition method did not appear to be required.

Clearly much additional research is needed to clarify these experimental results.

The authors thank Professor R. Chellappa for his comments and help in carrying out this research. A special word of thanks is due to Professors Bart Kosko and George Moore for convincing us of the potential power of the artificial neural network concept.

APPENDIX

Experiment #3 in Table 1 is used below to illustrate more concretely the capabilities and limitations of the linear associative memory approach.

We first give explicitly the 2×20 noise-free associative memory matrix $\hat{\mathbf{M}}^0$ for this example. As detailed in Section 3, this matrix was constructed using a selected training grid of 231 training parameter vectors $\mathbf{r}_i = (\omega_i, \phi_i)^T$ and a collection of 231 associated noise-free training stimulus vectors $\mathbf{s}_i$ generated as in (13) by means of the system equations (9).

Columns of $\hat{\mathbf{M}}^0$ (1–7):

6.2698	−6.2698	−29.4898	29.4898	71.7502	−71.7502	−116.4759
−3.8349	3.8349	13.7163	−13.7163	−30.8688	30.8688	48.2495

TABLE A.1

Parameter Estimates Obtained Using $\hat{M}^0$

Case	Test parameter vector $\mathbf{r}$	Estimated parameter vector $\hat{\mathbf{r}}$
(a)	$\omega = 0.41$	$\hat{\omega} = 0.42$
	$\phi = 0.23$	$\hat{\phi} = 0.24$
(b)	$\omega = 0.35$	$\hat{\omega} = 0.36$
	$\phi = 0.15$	$\hat{\phi} = 0.15$
(c)	$\omega = 1.1$	$\hat{\omega} = 0.9$
	$\phi = 0.6$	$\hat{\phi} = 0.6$
(d)	$\omega = 1.2$	$\hat{\omega} = 0.6$
	$\phi = 0.7$	$\hat{\phi} = 0.8$

Columns of $\hat{\mathbf{M}}^0$ (8–14):

116.4759	134.9118	−134.9118	−113.7151	113.7151	69.1102	−69.1102
−48.2495	−54.6267	54.6267	45.3331	−45.3331	−27.2346	27.2346

Columns of $\hat{\mathbf{M}}^0$ (15–20):

−29.0675	29.0675	7.6658	−7.6658	−0.9669	0.9669
11.3505	−11.3505	−2.9707	2.9707	0.3772	−0.3722

Using the matrix $\hat{\mathbf{M}}^0$, illustrative parameter estimates $\hat{\mathbf{r}}$ will now be presented in Table A.1 for the following four cases: (a) a test parameter vector $\mathbf{r}$ lying very close to a training parameter vector $\mathbf{r}_i$; (b) a test parameter vector $\mathbf{r}$ determined as a weighted average of four training parameter vectors; (c) a test parameter vector $\mathbf{r}$ lying just outside of the training grid; and (d) a test parameter vector $\mathbf{r}$ lying further outside of the training grid. Note the accuracy of the parameter estimates for cases (a) and (b). Cases (c) and (d) illustrate the deterioration in estimation accuracy which occurs as the test parameter vectors depart from the training grid.

A noisy associative memory matrix $\hat{\mathbf{M}}^{n*}$ for experiment #3 in Table 1 will next be presented. This matrix was constructed using the same training grid of 231 training parameter vectors $\mathbf{r}_i$ as was used for the noise-free matrix $\mathbf{M}^0$. However, the stimulus vectors $\mathbf{s}_i$ were corrupted by i.i.d. noise terms $\mathbf{n}_i$ with mutually independent components generated by means of a pseudorandom number generator for a zero-mean Gaussian distribution with standard deviation equal to 0.1. The columns of the stimulus matrix used in the construction of $\hat{\mathbf{M}}^{n*}$ then consisted of the noisy training stimulus vectors $\mathbf{s}_i^* = \mathbf{s}_i + \mathbf{n}_i$ in place of the noise-free training stimulus vectors $\mathbf{s}_i$.

Columns of $\hat{\mathbf{M}}^{n*}$ (1–7):

0.1728	−0.1622	−0.3873	0.3867	−0.1882	0.1697	0.1247
−0.6178	0.6394	0.1655	−0.2039	0.2617	−0.2341	−0.0078

Columns of $\hat{\mathbf{M}}^{n*}$ (8–14):

−0.0769	0.1008	−0.1769	0.0422	0.0249	−0.1068	0.0863
−0.0493	−0.0838	0.2188	−0.1120	0.0131	0.0240	0.0095

Columns of $\hat{\mathbf{M}}^{n}{}_{*}$ (15–20):

−0.0668	0.0222	0.1229	−0.0459	−0.0300	−0.0004
0.1786	−0.1499	−0.1108	0.0540	−0.0122	0.0302

Note that the components of the noisy associative memory matrix $\hat{\mathbf{M}}^{n*}$ are several orders of magnitude smaller than the components of the noise-free associative memory matrix $\hat{\mathbf{M}}^0$, as one would anticipate from the discussion in Section 4. Suppose the observation vectors are corrupted by the same kind of noise used in the construction of $\hat{\mathbf{M}}^{n*}$. Then, again from Section 4, one would anticipate that the parameter estimates obtained by use of $\hat{\mathbf{M}}^{n*}$ would have much greater precision than the parameter estimates obtained by use of $\hat{\mathbf{M}}^0$.

To illustrate this, we corrupt the components of the noise-free stimulus vector $\mathbf{s}$ for case (b) in Table A.1 with i.i.d. noise generated by means of a pseudorandom number generator for a zero-mean Gaussian distribution with standard deviation equal to 0.1. We then compare the parameter estimates in Table A.2 obtained for this noisy observation vector, first using the noise-free associative memory matrix $\hat{\mathbf{M}}^0$, and then using the noisy associative memory matrix $\hat{\mathbf{M}}^{n*}$.

The parameter estimates obtained using the noise-free associative memory matrix $\hat{\mathbf{M}}^0$ are so inaccurate as to be unusable. In contrast, the parameter estimates obtained using the noisy associative memory matrix $\hat{\mathbf{M}}^{n*}$ are accurate enough to provide usable preliminary estimates, e.g., for solving for $\mathbf{r}$ by a nonlinear least squares procedure.

TABLE A.2

Parameter Estimates Obtained for a Noisy Observation Vector

Test parameter vector $\mathbf{r}$	Estimate $\hat{\mathbf{r}}^0$ using $\hat{\mathbf{M}}^0$	Estimate $\mathbf{r}^*$ using $\hat{\mathbf{M}}^{n*}$
$\omega = 0.35$	$\hat{\omega}^0 = -31.30$	$\hat{\omega}^* = 0.43$
$\phi = 0.15$	$\hat{\phi}^0 = +13.76$	$\hat{\phi}^* = 0.08$

REFERENCES

1. T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, New York, 1988.
2. T. Poggio, On optimal nonlinear associative recall, *Biol. Cybernetics*, 19:201–209, 1975.
3. T. Poggio, Visual algorithms, in *Physical and Biological Processing of Images* (O. Braddick and A. Sleigh, Eds.), Springer-Verlag, 1983, New York, pp. 128–153.
4. P. A. Chou, The capacity of the Kanerva associative memory, *IEEE Trans. Inform. Theory*, IT-35:281–298 (Mar. 1989).
5. D. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, *Nature* 323:533–536 (Oct. 1986).
6. K. Hornik, M. Stinchcombe, and H. White, Multi-layer feedforward networks are universal approximators, Discussion Paper No. 88-45, Department of Economics, University of California, San Diego, June 1988.
7. B. Kosko, Bidirectional Associative Memories, *IEEE Trans. Syst. Man Cybernet.*, SMC-18: (Jan./Feb. 1988).
8. P. Olivier, Optimal noise rejection in linear associative memories, *IEEE Trans. Syst. Man Cybernet.* SMC-18:814–815 (Sep./Oct. 1988).
9. K. Murakami and T. Aibara, An improvement on the Moore–Penrose generalized inverse associative memory, *IEEE Trans. Syst. Man Cybernet.* SMC-17:699–706 (Aug. 1987).
10. L. Scales, *Introduction to Non-Linear Optimization*, Springer-Verlag, New York, 1985.
11. A. Albert, *Regression and the Moore–Penrose Pseudoinverse*, Academic, New York, 1972.

Received 25 August 1989; revised 22 September 1989