


UNIVERSITY OF
ILLINOIS LIBRARY
AT URBANA-CHAMPAIGN
BOOKSTACKS



Digitized by the Internet Archive
in 2011 with funding from
University of Illinois Urbana-Champaign

330
B385
1993:139 COPY 2

STX

THE LIBRARY OF THE

JUL 30 1993

UNIVERSITY OF ILLINOIS
LIBRARY COLLECTION

A RECURRENT NEWTON ALGORITHM AND ITS CONVERGENCE PROPERTIES

Chung-Ming Kuan
Department of Economics

BEBR

FACULTY WORKING PAPER NO. 93-0139

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

June 1993

A RECURRENT NEWTON ALGORITHM AND ITS CONVERGENCE PROPERTIES

Chung-Ming Kuan
Department of Economics

**A RECURRENT NEWTON ALGORITHM AND
ITS CONVERGENCE PROPERTIES**

Chung-Ming Kuan
Department of Economics
University of Illinois at Urbana-Champaign

June 17, 1993

† The author thanks Tung Liu for very useful comments and suggestions.

Abstract

In this paper a recurrent Newton algorithm for an important class of recurrent neural networks is introduced. It is noted that a suitable constraint must be imposed on recurrent variables to ensure proper convergence behavior. The simulation results show that the proposed Newton algorithm with the suggested constraint perform uniformly better than the back-propagation algorithm and the Newton algorithm without the constraint, in terms of mean-squared errors.

1 Introduction

It has been recognized that feedforward neural networks may have difficulty in representing certain sequential behavior of a target sequence, [1]. This deficiency hampers the applications of feedforward networks in the fields, such as signal processing and dynamic control, in which temporal structure plays an important role. Researchers are therefore motivated to study the so-called *recurrent* networks, e.g., [1]–[8]. A recurrent network can be obtained from a feedforward network by permitting additional, internal feedback channels, hence is capable of capturing more dynamic characteristics than does a feedforward network.

Owing to the existence of internal feedbacks in recurrent networks, learning algorithms for feedforward networks are not directly applicable. Kuan, Hornik, & White [9] propose a recurrent back-propagation (BP) algorithm and establish its almost sure convergence property under the condition that feedback connections are suitably constrained. Because the recurrent BP algorithm also performs a gradient search in the parameter space, it, as other gradient-based learning algorithms, converges very slowly. However, it is well known in system identification literature that a Newton algorithm is computationally and statistically more efficient than gradient-search algorithms. In this paper, we first introduce a recurrent Newton algorithm for an important class of recurrent networks and then sketch its almost sure convergence property. Similar to the recurrent BP algorithm, the recurrent variables must be constrained suitably in the recurrent Newton algorithm to ensure meaningful convergence. Our simulation results strongly indicate that the recurrent Newton algorithm with the suggested constraint yields better convergence results than the recurrent BP algorithm and the Newton algorithm without the constraint.

This paper is organized as follows. We first introduce a class of recurrent networks in section 2. The recurrent Newton algorithm and the constraint needed for recurrent variables are discussed in section 3. Simulation results are reported in section 4. The paper is concluded by Section 5. An example of the Newton algorithm is given in the Appendix.

2 Recurrent Network

Let O , H , X , and R denote column vectors of m network outputs, q hidden unit activations, n network inputs, and p internal feedbacks, respectively. The elements of these vectors are denoted using corresponding lower cases. At time t , a recurrent network with a single hidden layer and delayed internal feedbacks can be represented in the following generic form:

$$\begin{aligned} O_t &= \Phi(b + \beta' H_t), \\ H_t &= \Psi(c + \gamma' X_t + \delta' R_t), \end{aligned} \tag{1}$$

where Φ and Ψ are vector-valued functions, and

$$R_t = \Lambda(X_{t-1}, R_{t-1}; W), \tag{2}$$

with Λ also a vector-valued function and W the k -dimensional vector of network connection weights b , β , c , γ , and δ . If R_t is chosen to be O_{t-1} , (1) is a network with output feedbacks, Jordan [1]; if R_t is chosen to be H_{t-1} , it is a network with hidden-unit activations feedbacks, Elman [8]. When $R_t = 0$, this network simply reduces to a feedforward network. The fully-recurrent network of [7] can also be defined in a similar way with suitably defined R_t entering all the units. In this paper, however, we will confine ourselves to the class of recurrent networks described in (1) and (2).

Let Y_t denote the vector of m target variables. In a dynamic environment with time series data, important temporal structures are embedded in lagged targets Y_{t-1} , Y_{t-2} , etc. Thus, it is quite typical to use lagged targets as inputs for feedforward networks to capture dynamics. Clearly, networks with too few lagged targets will not be able to capture certain temporal structures that depend on a long history of targets. On the other hand, storing all the past information in memory is practically implausible. This situation is similar to building a linear AR (autoregressive) model in which a suitable number of AR lags is typically difficult to determine. This difficulty can be circumvented if a network has a “memory” device to store past information compactly. Recurrent networks by construction have this property. To see this, note that by recursive substitution,

$$\begin{aligned} R_t &= \Lambda(X_{t-1}, R_{t-1}; W) \\ &= \Lambda(X_{t-1}, \Lambda(X_{t-2}, R_{t-2}; W); W) \end{aligned}$$

$$\begin{aligned}
&= \vdots \\
&= \ell(X^{t-1}, W),
\end{aligned} \tag{3}$$

where $X^{t-1} = \{X_0, X_1, \dots, X_{t-1}\}$ is the collection of all previous inputs. As R_t depends on the entire history of inputs and all the connection weights in a complex manner, introducing recurrent variables to a feedforward network is somewhat similar to adding “moving average” terms to AR models. (In what follows, we also write R_t as $R_t(W)$ to signify its parameter dependence.) Thus, recurrent variables serve to summarize past information in a compact form, in terms of network outputs or hidden-unit activations. A recurrent network may therefore be interpreted as a parsimonious network model which incorporates all the past network inputs without storing all of them in memory. It is this property that makes recurrent networks attractive in dynamic applications.

From (1) and (2) we can write the output of a recurrent network as

$$\begin{aligned}
O_t &= \Phi(b + \beta' \Psi(c + \gamma' X_t + \delta' R_t(W))) \\
&=: F(X_t, R_t(W); W);
\end{aligned}$$

or by (3), $O_t =: f(X^t; W)$ is also a function of the entire history of network inputs. Given the mean-squared error (MSE) objective function, the parameters of interest are W^* which minimize

$$\begin{aligned}
&\lim_{t \rightarrow \infty} \mathbb{E}(Y_t - f(X^t; W))^2 \\
&= \lim_{t \rightarrow \infty} \mathbb{E}(Y_t - \mathbb{E}(Y_t|X^t))^2 + \lim_{t \rightarrow \infty} \mathbb{E}(\mathbb{E}(Y_t|X^t) - f(X^t; W))^2;
\end{aligned} \tag{4}$$

here, the limit is taken to permit system feedbacks. Observe that the first term on the right-hand side of (4) is an intrinsic error which does *not* depend on W . Hence, W^* minimizes the limit of the approximation error: $\mathbb{E}(\mathbb{E}(Y_t|X^t) - f(X^t; W))^2$ in (4). It is well known that $\mathbb{E}(Y_t|X^t)$ is the best L_2 -predictor of Y_t given the σ -algebra generated by X^t , denoted as $\sigma(X^t)$. As $\mathbb{E}(Y_t|X_t)$ is measurable with respect to $\sigma(X^t)$,

$$\mathbb{E}(Y_t - \mathbb{E}(Y_t|X^t))^2 \leq \mathbb{E}(Y_t - \mathbb{E}(Y_t|X_t))^2.$$

Therefore, a recurrent network may characterize the behavior of Y_t better by approximating the conditional mean function $\mathbb{E}(Y_t|X^t)$, whereas a feedforward network with input X_t can only approximate $\mathbb{E}(Y_t|X_t)$.

3 A Recurrent Newton Algorithm

It should be clear that any learning algorithm (on-line or off-line) obtained from the objective function (4) must take into account the fact that $R_t(W)$ depends on network connection weights. Let E_t be the network error, i.e., $E_t = Y_t - F(X_t, R_t(W); W)$, which also depends on W directly and indirectly through the presence of $R_t(W)$. The derivatives of E with respect to W are, by chain rule,

$$\nabla E_t = - \underbrace{F_W(X_t, R_t(W); W)}_{k \times m} - \underbrace{\nabla R_t(W)}_{k \times p} \underbrace{F_R(X_t, R_t(W); W)}_{p \times m},$$

where F_W and F_R are matrices of the first order derivatives of F with respect to W and R , respectively. Because the BP algorithm for feedforward networks contains only the term F_W , it is clear that it does not follow a correct gradient direction when recurrent variables $R_t(W)$ are present. Therefore, it is extremely important for an algorithm in recurrent networks to incorporate the additional term, $\nabla R_t(W) F_R$, so as to maintain a correct search direction. A learning algorithm that ignores this term need not converge to a MSE minimizer. In light of (3), it is evident that computation of $\nabla R_t(W)$ requires all the past inputs. Hence, such computation becomes practically formidable because all the past inputs must be stored in memory and because computation will increase with t . This problem can be circumvented by recursively approximating $\nabla R_t(W)$ via

$$\nabla R_t(W) = \underbrace{\Lambda_W(X_{t-1}, R_{t-1}(W); W)}_{k \times p} + \underbrace{\nabla R_{t-1}(W)}_{k \times p} \underbrace{\Lambda_R(X_{t-1}, R_{t-1}(W); W)}_{p \times p},$$

where Λ_W and Λ_R are matrices of the first order derivatives of Λ with respect to W and R , respectively. This observation motivates the recurrent BP algorithm studied in [9].

As a gradient descent algorithm, the BP algorithm for feedforward networks converges very slowly and is statistically inefficient, see e.g., [10]. The recurrent BP algorithm therefore has the same disadvantage. In numerical optimization, better convergence results can be obtained from the Newton method with a search direction based on the second-order derivatives (the Hessian matrix). When the objective function is quadratic, the Newton method converges to the minimum of the objective function in one iteration. To ensure that the search direction always points “downhill”, it is also typical to use a positive-definite matrix, such as the outer product of the gradient vector, to approximate the Hessian

matrix. In view of this, a natural extension of the recurrent BP algorithm is an algorithm analogous to the Newton method in numerical optimization. This type of algorithms is well known in system identification literature, e.g., Ljung & Söderström [11]. It is shown in [10] that the stochastic Newton learning algorithm for feedforward networks is computationally and statistically more efficient than the BP algorithm; in particular, it is asymptotically equivalent to the nonlinear least squares estimator under very general conditions. In what follows, a variable is written with the “hat” symbol if it is evaluated at parameter estimates. Bearing the issue of taking gradients correctly in mind, a straightforward Newton algorithm is as follows.

$$\hat{E}_t = Y_t - F(X_t, \hat{R}_t; \hat{W}_t), \quad (5)$$

$$\nabla \hat{E}_t = -F_W(X_t, \hat{R}_t; \hat{W}_t) - \hat{D}_t F_R(X_t, \hat{R}_t; \hat{W}_t), \quad (6)$$

$$\hat{W}_{t+1} = \hat{W}_t - \eta_t \hat{G}_{t+1}^{-1} (\nabla \hat{E}_t \hat{E}_t), \quad (7)$$

$$\hat{G}_{t+1} = \hat{G}_t + \eta_t (\nabla \hat{E}_t \nabla \hat{E}_t' - \hat{G}_t), \quad (8)$$

$$\hat{R}_{t+1} = \Lambda(X_t, \hat{R}_t; \hat{W}_t), \quad (9)$$

$$\hat{D}_{t+1} = \Lambda_W(X_t, \hat{R}_t; \hat{W}_t) + \hat{D}_t \Lambda_R(X_t, \hat{R}_t; \hat{W}_t). \quad (10)$$

where $\{\eta_t\}$ is a sequence of learning rates of order $1/t$, and \hat{D}_t is used in lieu of $\nabla \hat{R}_t$. Clearly, if R is not present in the network, a recurrent network is just a feedforward network, and the recurrent Newton algorithm simply reduces to the Newton algorithm for feedforward networks, [10]. In contrast with the recurrent BP algorithm, the recurrent Newton algorithm contains an additional updating equation (8) which recursively updates the outer product of $\nabla \hat{E}_t$ so as to provide an approximate Newton direction for (7).

There are some basic difficulties associated with the proposed algorithm. First, (7) involves matrix inversion which is not desirable in a recursive algorithm. Second, \hat{G}_t must be a positive definite matrix to assure correct search direction. In practice, some modifications are needed to avoid these difficulties. Let $\hat{P}_{t+1} = \eta_t \hat{G}_{t+1}^{-1}$ and $\nu_t = (1 - \eta_t)\eta_{t-1}/\eta_t$. Then by a matrix inversion formula,

$$\hat{P}_{t+1} = \frac{1}{\nu_t} \left(\hat{P}_t - \hat{P}_t \nabla \hat{E}_t (\nabla \hat{E}_t' \hat{P}_t \nabla \hat{E}_t + \nu_t)^{-1} \nabla \hat{E}_t' \hat{P}_t \right).$$

For the network with a single output (i.e., $m = 1$) so that $\nabla \hat{E}_t$ is $k \times 1$,

$$\hat{P}_{t+1} = \frac{1}{\nu_t} \left(\hat{P}_t - \frac{\hat{P}_t \nabla \hat{E}_t \nabla \hat{E}_t' \hat{P}_t}{\nabla \hat{E}_t' \hat{P}_t \nabla \hat{E}_t + \nu_t} \right), \quad (11)$$

which does not involve matrix inversion. For the network with multiple outputs (i.e., $m > 1$), we follow the approach of Bierman [12] and compute \hat{P}_t using a sequence of single-output algorithms in which no matrix inversion is needed. This leads to a modified algorithm which contains (5), (6), (9), and (10) in the original version but substitutes the updating equations below for (7) and (8):

$$\hat{W}_{t+1} = \hat{W}_t - \hat{P}_{t+1}(\nabla \hat{E}_t \hat{E}_t), \quad (12)$$

$$\bar{P}_{t+1}^{(0)} = \hat{P}_t, \quad (13)$$

$$\bar{P}_{t+1}^{(j)} = \bar{P}_{t+1}^{(j-1)} - \frac{\bar{P}_{t+1}^{(j-1)} \nabla \hat{E}_{j,t} \nabla \hat{E}'_{j,t} \bar{P}_{t+1}^{(j-1)}}{\nabla \hat{E}'_{j,t} \bar{P}_{t+1}^{(j-1)} \nabla \hat{E}_{j,t} + \nu_t}, \quad j = 1, \dots, m, \quad (14)$$

$$\bar{P}_{t+1} = \bar{P}_{t+1}^{(m)} / \nu_t, \quad (15)$$

$$\hat{P}_{t+1} = \begin{cases} \bar{P}_{t+1}, & \text{if } \bar{P}_{t+1} - \epsilon I_k \text{ is p.s.d.}, \\ \bar{P}_{t+1} + M_{t+1}, & \text{otherwise,} \end{cases} \quad (16)$$

where $\nabla \hat{E}_{j,t}$ is the j -th column of $\nabla \hat{E}_t$, “p.s.d” stands for positive semidefinite, ϵ is a small positive constant, and M_{t+1} is chosen to make $\hat{P}_{t+1} - \epsilon I_k$ a p.s.d. matrix. Note that in (13)–(15), \bar{P}_{t+1} is updated as each output unit is added sequentially, and that (16) implements a correction ensuring \hat{P}_t to be a p.s.d. matrix. This modified version is analogous to the Newton algorithm considered in system identification literature; for more details and related numerical issues of the Newton algorithm we refer to [11, Chap. 6]. A simple example of this modified algorithm is given in the Appendix.

Let $\theta := [W' (\text{vec } P)']'$ be the s -dimensional column vector of parameters, where the vec operator stacks all columns of a matrix into a column vector, and $\hat{\theta}_t$ be its estimate. To prevent $\hat{\theta}_t$ from diverging to infinity, it is also standard to impose a projection device π on these estimates. Given a compact parameter space Θ , if $\hat{\theta}_t \in \Theta$, $\pi(\hat{\theta}_t) = \hat{\theta}_t$; if $\hat{\theta}_t \notin \Theta$, $\pi(\hat{\theta}_t)$ takes a value in Θ . The recurrent Newton algorithm proposed in this paper is the modified algorithm discussed above together with a *truncation* device on $\hat{\theta}_t$. In practice, we choose large truncation bounds for \hat{b}_t , $\hat{\beta}_t$, \hat{c}_t , $\hat{\gamma}_t$, and \hat{P}_t so that the behavior of these estimates is virtually not affected; to ensure proper convergence behavior, however, some restrictive bounds on the estimates of recurrent connection weights, $\hat{\delta}_t$, are needed. These bounds are discussed below.

The almost sure convergence property of $\hat{\theta}_t$ can be proved by combining the results in Kuan & White [10, 13], which are based on the compactness approach of the ODE

(ordinary differential equation) method due to [14]; see also [9]. To reduce technicality, we do not provide a formal theorem but only sketch this convergence property. Write

$$Z_t(\theta) = [Y_t' \ X_t' \ R_t(\theta)' \ (\text{vec } D_t(\theta))']'$$

and $\hat{Z}_t = Z_t(\hat{\theta}_t)$. Then the updating equations (12)–(16) can be written compactly as

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \eta_t Q(\hat{Z}_t; \hat{\theta}_t). \quad (17)$$

Let

$$\bar{Q}(\theta) = \lim_{t \rightarrow \infty} \mathbb{E}[Q(Z_t(\theta); \theta)];$$

note that it is just the first order condition of (4). Under very general conditions on the data Y_t and X_t and network functions Φ and Ψ , if the recurrent function Λ is a contraction mapping in R (i.e., for each x and θ , $|\Lambda_R(x, \cdot; \theta)| \leq \alpha_0 < 1$), it can be shown that $\hat{\theta}_t$ eventually behave like the solution path of the ODE $\dot{\theta} = \bar{Q}(\theta)$ and converge with probability one to the set of all locally asymptotically stable equilibria in Θ for this ODE. If this set contains only finitely many point, $\hat{\theta}_t$ converges to one locally asymptotically stable equilibrium, hence a local minimum of (4).

Typically, the aforementioned convergence property holds when Φ and Ψ are continuously differentiable of order two; most of commonly used network functions, such as the logistic or hyperbolic tangent functions, satisfy this property. On the other hand, the contraction mapping condition on Λ is crucial and is *not* satisfied automatically. In view of (3), R_t could “explode” if Λ is not a contraction mapping, because the effects of x_t would accumulate very rapidly. Even when Λ is a bounded (or squashing) function, this condition is still needed; otherwise, R_t would be approaching to the upper or lower bound of Λ in a short learning period. For example, given an Elman network so that $\Lambda = \Psi$, if Ψ is the logistic function, then hidden unit activations would be close to zero or one if Ψ is not a contraction mapping in lagged hidden-unit activations. This causes “exaggeration” of the behavior of hidden units and invalidates the learning results. If Λ is a contraction mapping, the recurrent variables in fact implement an exponentially forgetting memory of the data sequence and are well behaved essentially. (We note that the contraction mapping requirement of Λ is similar to the “invertibility” condition for time series models

with moving average terms.) Let $M_\Phi = \sup_e \Phi'(e)$ and $M_\Psi = \sup_u \Psi'(u)$. From [9], the contraction mapping property for the Jordan network is satisfied when

$$\sum_{j=1}^q |\beta_j| |\delta_j| < (M_\Phi M_\Psi)^{-1},$$

where $|\cdot|$ stands for Euclidean norm, and for the Elman network, it is satisfied when

$$\sum_{i=1}^q \delta'_i \delta_i < (M_\Psi)^{-2}.$$

That is, the connection weights must be suitably constrained during the learning process so as to ensure proper convergence behavior. Note that in the Jordan network the connection weights β 's must be restricted; hence the representability of the Jordan network is unavoidably affected by this constraint. In the Elman network, however, only recurrent connections are subject to the constraint so that feedforward part of the network is not affected. Thus, as far as the representation capability of a network is concerned, the Elman network seems to be more desirable since less network connection weights are restricted. It is straightforward to verify that some sufficient conditions ensuring the contraction mapping property in the Elman network are that $|\delta_{ij}| < 4/q$ for all i and j if Ψ is the logistic function and that $|\delta_{ij}| < 1/q$ for all i and j if Ψ is the hyperbolic tangent function. We stress that such restrictions are not only of theoretical interest but also of practical importance, as shown in the simulation results below.

4 Simulations

To evaluate the performance of the proposed Newton algorithm, we conduct the following simulations. The target variables y_t , $t = 1, \dots, T$, are generated from three models: (1) a bilinear model:

$$y_t = 0.4y_{t-1} - 0.3y_{t-2} + 0.5y_{t-1}\epsilon_{t-1} + \epsilon_t,$$

where ϵ_t are independent $N(0, 1)$; (2) a Hénon map:

$$\begin{aligned} x_t &= 0.3y_{t-1}, \\ y_t &= 1 + x_{t-1} - 1.4y_{t-1}^2, \end{aligned}$$

where $y_0 = -1 \times u$ and $y_{-1} = 0.5 \times u$, u is the uniform random variable on $[0, 1]$; (3) a SETAR (self-exciting threshold autoregressive) model:

$$y_t = \begin{cases} 0.9y_{t-1} + \epsilon_t, & |y_{t-1}| \leq 1, \\ -0.3y_{t-1} + \epsilon_t, & |y_{t-1}| > 1, \end{cases}$$

where ϵ_t are independent $N(0, 1)$. In the first two models y_t depend on its own past values; in the SETAR model y_t depends only on y_{t-1} . We include the third model to see how the algorithms perform when a recurrent network is not really needed. In the simulations, the sample size T is 1000, and the number of replications is 200. The network inputs are lagged target variables y_{t-1} and y_{t-2} . The network activation function Φ is the identity function and Ψ is the logistic function. We estimate the Elman network with 4–6 hidden units using four algorithms: the BP and Newton algorithms, each with and without the constraint $|\delta_{ij}| \leq 3.995/q$, where q is the number of hidden units. The initial feedforward connection weights (β 's and γ 's) are generated from $N(0, 1)$ and recurrent connection weights (δ 's) are generated from $10 \times N(0, 1)$. This allows us to assess the effectiveness of the proposed constraint more easily.

In the simulation MSE at each recursive step is recorded and averaged over 200 replications. The averages of MSE's from the last 500 recursive steps and the last MSE's in the final (1000-th) recursive step are summarized in Table 1. We observe from this table that:

1. For all cases considered, the Newton algorithm with the suggested constraint yields lowest average and last MSE, and the BP algorithm without the constraint yields the highest average and last MSE. The Newton algorithm without the constraint performs even better than the recurrent BP algorithm with the constraint.
2. The average and last MSE of the (Newton and BP) algorithms without the constraint may be increasing with the number of hidden units. Except for the SETAR model, the average and last MSE of the algorithms with the constraint decreases with the number of hidden units.

The first result shows that the Newton algorithm with the suggested constraint performs uniformly better than the other three algorithms in terms of MSE's. It is also interesting to note from the second result that adding more hidden units need not result in lower MSE

if a learning algorithms is used without the constraint. In the SETAR model, a recurrent network is not really needed; hence the Newton algorithm with the constraint results in similar final MSE regardless of the number of hidden units. However, the MSE of the BP algorithm with the constraint actually increases with the number of hidden units. We also observe from the simulation results that, after some recursive steps, the MSE's of the four algorithms have the following relationship:

- Newton with the constraint
- < Newton without the constraint
- < BP with the constraint
- < BP without the constraint.

To conserve space, we only plot those MSE's for the networks with 6 hidden units in Figures 1-3; the MSE's of the BP algorithm without the constraint are not included in the figures because they are too large relative to MSE's from other algorithms. From these figures, it can be seen that the Newton algorithm without the constraint behaves unstably and may produce very large errors during the learning period. It can also be seen that both the Newton and BP algorithms with the constraint are well behaved, but the Newton algorithm results in much lower MSE and converges much quickly than does the BP algorithm. These results clearly show the superiority of the proposed algorithm.

5 Conclusions

In this paper we propose a recurrent Newton algorithm which extends the recurrent BP algorithm introduced earlier to allow for a Newton search in the parameter space. To ensure proper convergence behavior, a constraint must be imposed to prevent recurrent variables from "exploding". The simulation results demonstrate that the proposed algorithm with the constraint performs uniformly better than other algorithms in terms of MSE.

Appendix

An Example of the Recurrent Newton Algorithm: For notational simplicity, we consider a single-output Elman network with Φ the identity function and Ψ the logistic function:

$$\begin{aligned} O_t &= b + \beta' H_t = b + \sum_{i=1}^q \beta_i h_{it}, \\ h_{it} &= \frac{1}{1 + \exp(-c_i - \gamma'_i X_t - \delta'_i H_{t-1})}. \end{aligned}$$

In this case, $R = H$, $\Lambda = \Psi$, and

$$W = [b \ \beta' \ c_1 \ \gamma'_1 \ \dots \ c_q \ \gamma'_q \ \delta'_1 \ \dots \ \delta'_q]'$$

The updating equations (12)–(16) can be easily computed from the following information.

Let $\tilde{X}_t = [1 \ X_t']'$, $\tilde{\beta} = [b \ \beta']'$, and $\tilde{\gamma}_i = [c_i \ \gamma'_i]'$. The network error in (5) is computed as

$$\begin{aligned} \hat{E}_t &= Y_t - (\hat{b}_t + \sum_{i=1}^q \hat{\beta}_i \hat{h}_{it}), \\ \hat{h}_{it} &= \frac{1}{1 + \exp(-\hat{c}_{it} - \hat{\gamma}'_{it} X_t - \hat{\delta}'_{it} \hat{H}_{t-1})}. \end{aligned}$$

In (6), the vector F_W contains the following sub-vectors:

$$\begin{aligned} F_{\tilde{\beta}} &= [1 \ \hat{H}'_t]', \\ F_{\tilde{\gamma}_i} &= \hat{\beta}_i \hat{h}_{it} (1 - \hat{h}_{it}) \tilde{X}_t, \\ F_{\delta_i} &= \hat{\beta}_i \hat{h}_{it} (1 - \hat{h}_{it}) \hat{H}_{t-1}, \end{aligned}$$

for $i = 1, \dots, q$; and the vector F_H is

$$F_H = \sum_{i=1}^q \hat{\beta}_i \hat{h}_{it} (1 - \hat{h}_{it}) \delta_{it}.$$

The recurrent variables of (9) are \hat{h}_{it} . In (10), the matrix Ψ_W contains the following submatrices:

$$\begin{aligned} \Psi_{\tilde{\beta}} &= 0, \\ \Psi_{\tilde{\gamma}} &= \text{diag}[\hat{h}_{1t}(1 - \hat{h}_{1t})\tilde{X}_t \ \dots \ \hat{h}_{qt}(1 - \hat{h}_{qt})\tilde{X}_t], \\ \Psi_{\delta} &= \text{diag}[\hat{h}_{1t}(1 - \hat{h}_{1t})\hat{H}_{t-1} \ \dots \ \hat{h}_{qt}(1 - \hat{h}_{qt})\hat{H}_{t-1}]; \end{aligned}$$

and the matrix Ψ_H is

$$\Psi_H = [\hat{h}_{1t}(1 - \hat{h}_{1t})\delta_{1t} \ \dots \ \hat{h}_{qt}(1 - \hat{h}_{qt})\delta_{qt}].$$

The following values may be used to initialize the algorithm: the elements of \hat{W}_1 are randomly generated from some random number generator, $\hat{G}_1 = sI$ with $s = 100/(\sum_t y_t^2/T)$ and I the identity matrix, $\hat{h}_{i1} = 1/2$ for all i , and $\hat{D}_1 = 0$.

References

- [1] M. Jordan, "Serial order: A parallel distributed processing approach," ICS Report 8604, Institute for Cognitive Science, University of California, San Diego, 1986.
- [2] M. Jordan, "Constrained supervised learning," *Journal of Mathematical Psychology*, vol. 36, pp. 396–425, 1992.
- [3] L. B. Almeida, "A learning rule for asynchronous perceptrons with feedback in a combinatorial environment," in *Proceedings of the IEEE First International Conference on Neural Networks*, 1987, pp. II: 609–618.
- [4] F. J. Pineda, "Generalization of back-propagation to recurrent neural networks," *Physical Review Letters*, vol. 59, pp. 2229–2232, 1987.
- [5] M. Gherrity, "A learning algorithm for analog, fully recurrent neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, 1989, pp. I: 643–644,
- [6] B. A. Pearlmutter, "Learning state space trajectories in recurrent neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, 1989, pp. II: 365–372,
- [7] R. J. Williams and C. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270–280, 1989.
- [8] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.
- [9] C.-M. Kuan, K. Hornik and H. White, "A convergence result for learning in recurrent neural networks," *Neural Computation*, forthcoming, 1993.
- [10] C.-M. Kuan and H. White, Artificial neural networks: An econometric perspective, *Econometric Reviews*, forthcoming, 1993.
- [11] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*, Cambridge, MA: MIT Press, 1983.

- [12] G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation*, New York: Academic Press, 1977.
- [13] C.-M. Kuan and H. White, "Adaptive learning with nonlinear dynamics driven by dependent processes," Working Paper, Department of Economics, University of California, San Diego, 1993.
- [14] H. J. Kushner and D. S. Clark, *Stochastic Approximation Methods for Constrained and Unconstrained Systems*, New York: Springer-Verlag, 1978.

Table 1. Summary of Simulation Results.

Model	Hidden Units	Newton with Const.		Newton w/o Const.		BP with Const.		BP w/o Const.	
		Average MSE	Last MSE	Average MSE	Last MSE	Average MSE	Last MSE	Average MSE	Last MSE
Bi- linear	4	1.825	1.824	1.900	1.839	2.181	2.154	2.547	2.533
	5	1.820	1.811	1.902	1.852	2.139	2.111	2.579	2.564
	6	1.802	1.789	1.924	1.867	2.109	2.078	2.634	2.615
Hénon Map	4	0.125	0.125	0.176	0.162	0.329	0.324	0.491	0.488
	5	0.101	0.098	0.159	0.142	0.324	0.319	0.527	0.524
	6	0.079	0.079	0.152	0.158	0.302	0.296	0.536	0.530
SETAR	4	1.014	1.011	1.038	1.032	1.059	1.054	1.115	1.114
	5	1.015	1.011	1.037	1.030	1.066	1.060	1.118	1.114
	6	1.017	1.012	1.042	1.033	1.072	1.066	1.191	1.182

Figure 1. Bilinear Model: Elman Network with 6 Hidden Units

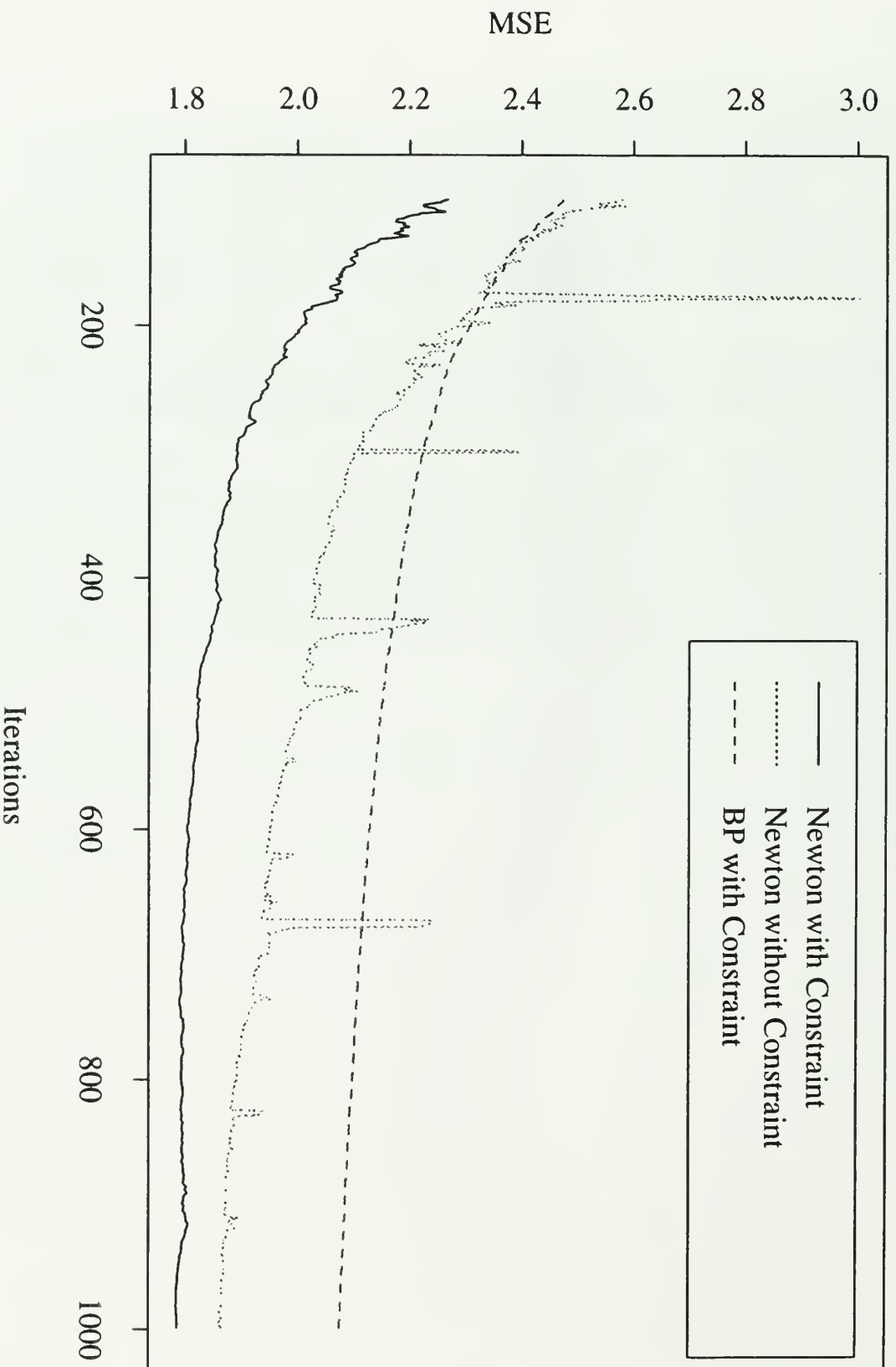


Figure 2. Henon Map Model: Elman Network with 6 Hidden Units

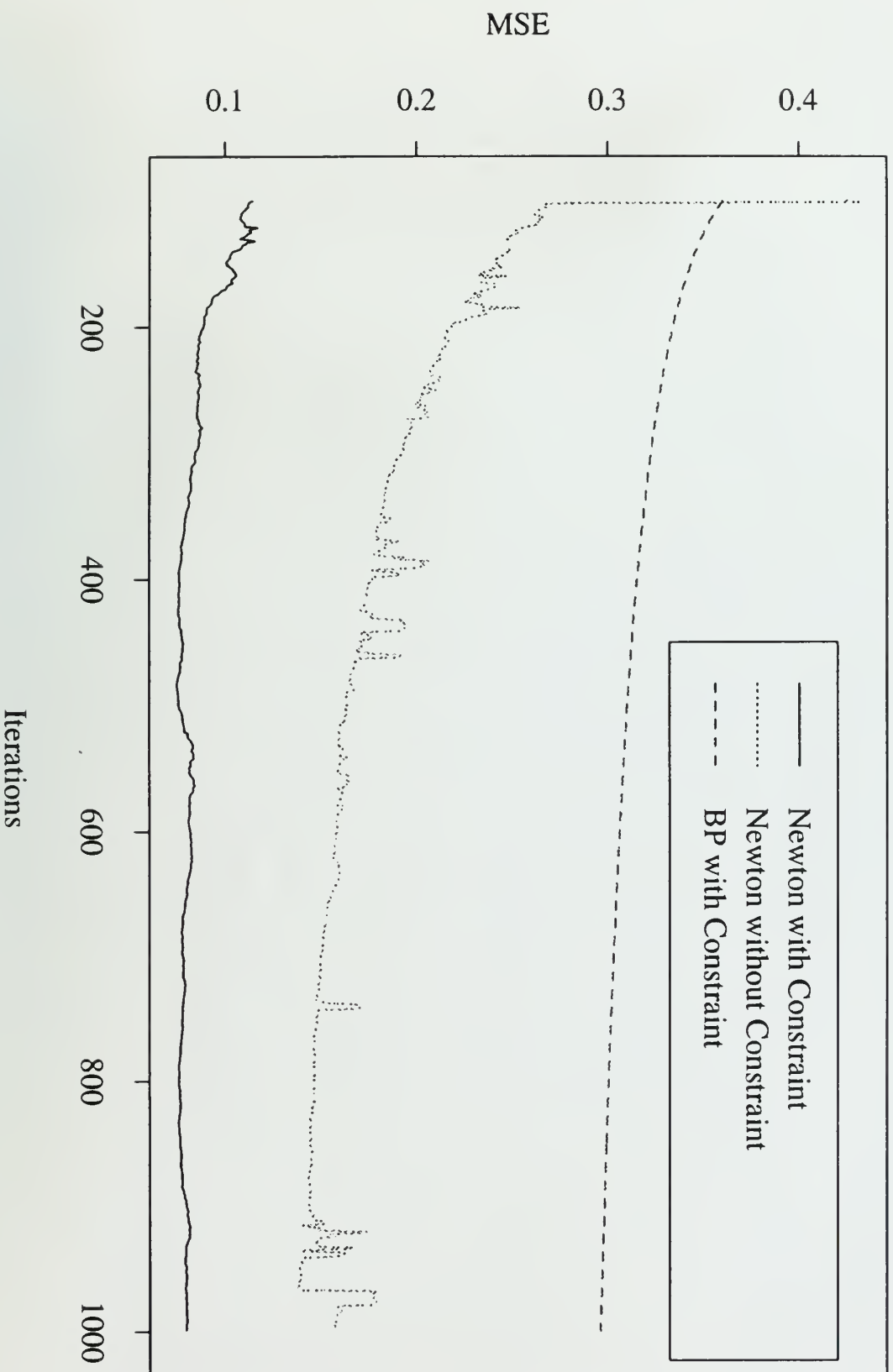
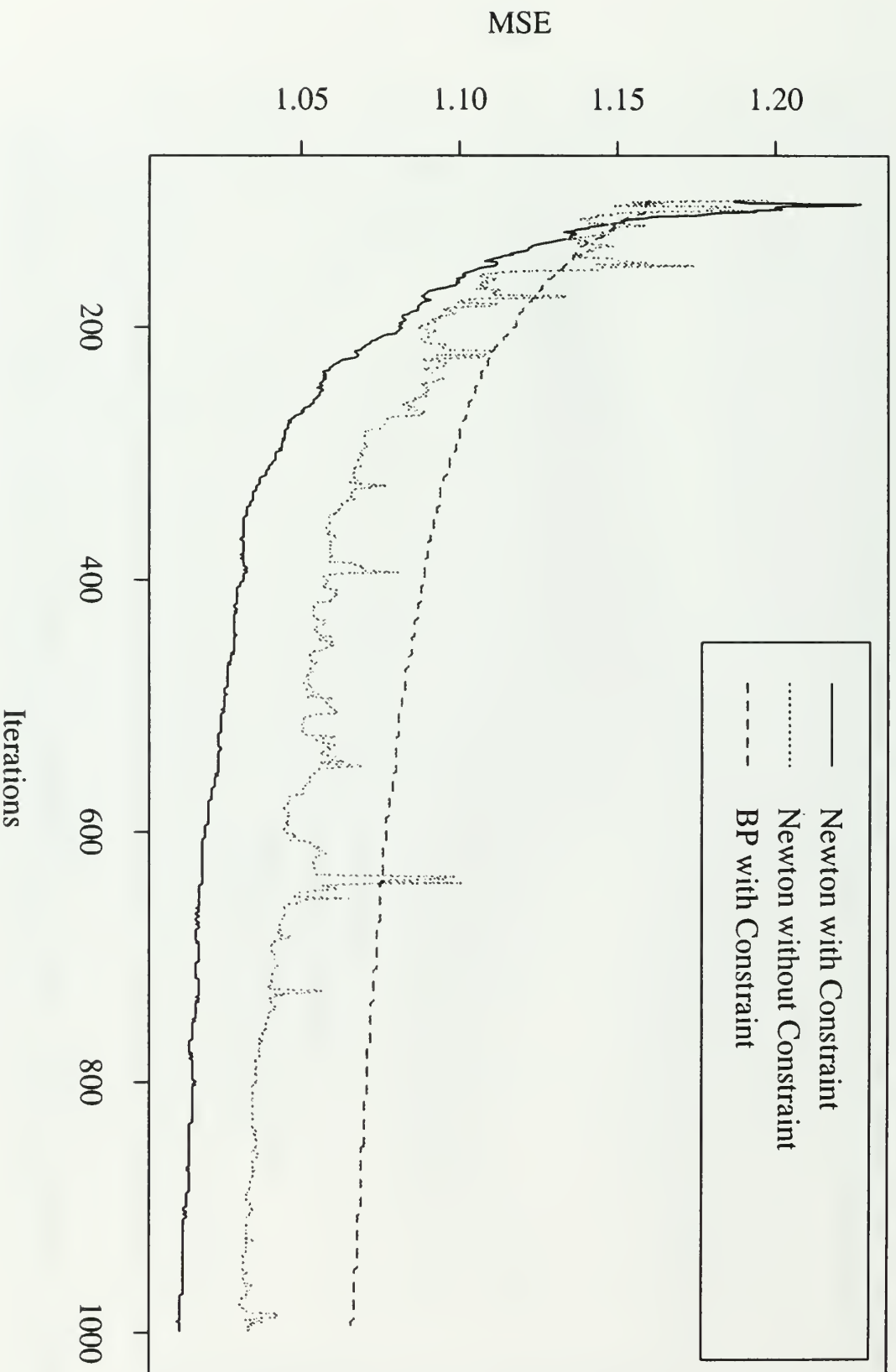


Figure 3. SETAR Model: Elman Network with 6 Hidden Units



HECKMAN
BINDERY INC.



JUN 95

Bound-To-Please® N. MANCHESTER,
INDIANA 46962

UNIVERSITY OF ILLINOIS-URBANA



3 0112 046973563