Nonlinear Modelling and Prediction with Feedforward and Recurrent Networks

Ramazan Gençay*
Department of Economics
University of Windsor
Canada

Tung Liu
Department of Economics
Ball State University
U.S.A.

November 1996

Abstract

In feedforward networks, signals flow in only one direction without feedback. Applications in forecasting, signal processing and control require explicit treatment of dynamics. Feedforward networks can accommodate dynamics by including past input and target values in an augmented set of inputs. A much richer dynamic representation results from also allowing for internal network feedbacks. These types of network models are called recurrent network models and are used by Jordan (1986) for controlling and learning smooth robot movements, and by Elman (1990) for learning and representing temporal structure in linguistics. In Jordan's network, past values of network output feed back into hidden units; in Elman's network, past values of hidden units feed back into themselves.

The main focus of this study is to investigate the relative forecast performance of the Elman type recurrent network models in comparison to feedforward networks with deterministic and noisy data. The salient property of the Elman type recurrent network architecture is that the hidden unit activation functions (internal states) are fed back at every time step to provide an *additional input*. This recurrence gives the network dynamical properties which make it possible for the network to have internal memory. Exactly *how* this memory is represented in the internal states is not determined in advance. Instead, the network must discover the underlying temporal structure of the task and learn to encode that structure internally. The simulation results of this paper indicate that recurrent networks filter noise more successfully than feedforward networks in small as well as large samples.

*Ramazan Gençay thanks the Natural Sciences and Engineering Research Council of Canada and the Social Sciences and Humanities Research Council of Canada for financial support.

1. Introduction

The standard problem in dynamical system analysis involves the description of the asymptotic behavior of the iterates of a given nonlinear system. The inverse problem, on the other hand, involves the construction of a nonlinear map from a sequence of its iterates. The constructed map can then be a candidate for a predictive model. Consider a dynamical system, $f : R^n \to R^n$, with the trajectory

$$x_{t+1} = f(x_t), \qquad t = 0, 1, 2, \ldots. \tag{1}$$

In practice, one rarely has the advantage of observing the true state of the system, let alone knowing the actual functional form, f, which generates the dynamics. The model that is used is the following: associated with the dynamical system in (1) there is a measurement function $g : R^n \to R^n$ which generates observations

$$y_t = g(x_t). \tag{2}$$

It is assumed that only the sequence $\{y_t\}$ is available to reconstruct f. Under certain regularity conditions, the Takens (1981) and Mañé (1981) embedding theorems indicate that this is feasible.

There are a variety of numerical techniques for modelling and prediction of nonlinear time series such as the threshold model of Tong (1983); exponential model of Ozaki (1982); local linear model of Farmer and Sidorowich (1987, 1988); nearest neighbors regression of Yakowitz (1987) and Stengos (1995); feedforward network model of Lapedes and Farber (1987a, b) and Gençay (1994). In addition, the Taylor series expansion, radial basis function of Casdagli (1989) and the nonparametric kernel regression are also used for nonlinear prediction. These techniques essentially involve interpolating or approximating unknown functions from scattered data points. The idea behind the Taylor series expansion is to increase the order of the expansion to the point where a curved surface of that order can follow the curvature of the local data points closely. The trade off with this method is that the number of terms in a multidimensional Taylor series expansion increases quite rapidly with the order. Indeed, the number of parameters needed for a Taylor series of a given order grows multiplicatively as the order of the expansion is increased and this method involves a choice of an optimal order of expansion. Casdagli (1989) points out that there are no known order of convergence results for $n > 1$, and that polynomials of high degree have an undesirable tendency to oscillate wildly.

The nonparametric kernel estimation is a method for estimating probability density functions from observed data. It is a generalization of histograms to continuously differentiable density estimators. The kernel density estimation involves the choice of a kernel function and a smoothing parameter. The idea behind this method is to determine the influence of each data point by placing a weight to each of the data points. The kernel function determines the shape of these weights and the window width determines their width. The approximation of an unknown function from the data can be obtained by calculating the conditional mean of the regression function. The kernel density estimator works in regression models with a

few lags. However, as the number of lags gets larger the rate of convergence of the nonparametric kernel density estimator slows down considerably, which leads to the deterioration of the estimator of the conditional mean in finite samples. There is further deterioration in the partial derivatives of the conditional mean estimator.

Radial basis functions are related to the kernel density estimator. In radial basis functions the contribution of each point is computed by least squares and these functions are easy to implement numerically. If standard algorithms for the solution of linear systems of equations are used, Casdagli (1989) indicates that for large data sets, their implementation is no longer feasible on standard workstations.

Among these techniques, artificial neural networks is one of the most recent techniques used in the nonlinear signal processing problem. This is partly due to some modelling problems encountered in the early stage of development. Earliest applications of feedforward networks are analyzed in Lapedes and Farber (1987a,b). Recent developments in the artificial neural network literature however, have provided the theoretical foundations for the universality of feedforward networks as function approximators. The results in Cybenko (1989), Funahashi (1989), Hornik, Stinchcombe and White (1989, 1990), and Hornik (1991) indicate that feedforward networks with sufficiently many hidden units and properly adjusted parameters can approximate an arbitrary function arbitrarily well in useful spaces of functions. Hornik, Stinchcombe, and White (1990) and Hornik (1991) further show that the feedforward networks can also approximate the derivatives of an arbitrary function. The advantages of these network models over other methods mentioned above are that feedforward network models use only linearly *many* parameters $O(qn)$, whereas traditional polynomial, spline, and trigonometric expansions use exponentially *many* parameters $O(q^n)$ to achieve the same approximation rate (Barron (1991)). A recent survey of this literature is presented in Kuan and White (1994).

In feedforward networks, signals flow in only one direction, without feedback. Applications in forecasting, signal processing and control require explicit treatment of dynamics. Feedforward networks can accommodate dynamics by including past input and target values in an augmented set of inputs. A much richer dynamic representation results from also allowing for internal network feedbacks. These types of network models are called recurrent network models and are used by Jordan (1986) for controlling and learning smooth robot movements, and by Elman (1990) for learning and representing temporal structure in linguistic. In Jordan's network, past values of network output feed back into hidden units; in Elman's network, past values of hidden units feed back into themselves.

The main focus of this study is to investigate the relative forecast performance of the Elman type of recurrent network models in comparison to the feedforward networks. The first stage of this study focuses on deterministic nonlinear time series estimation. The quality of the results with deterministic data will serve as a benchmark performance of the estimation techniques under study. The second stage involves the investigation of out-of-sample performances of the recurrent and feedforward network models with noisy data sets. The noise

component is investigated as a measurement noise. The out-of-sample mean square errors are used as the criteria for the quality of the forecasts.

The objective of this paper is to provide an informative comparison of the feedforward and recurrent networks within the context of nonlinear signal processing with noisy time series data. The results of this paper indicate that recurrent networks filter noise more successfully than feedforward networks in small as well as large samples. This suggests that recurrent networks act as more effective filters in the analysis of nonlinear dynamics from noisy time series data. Feedforward and recurrent networks are reviewed in Section 2. Simulation design, including the descriptions of the simulated model, estimation and forecast approach, and comparison statistics, are introduced in Section 3. Section 4 presents numerical results. We conclude thereafter.

2. Feedforward and Recurrent Networks

Over the past decade, researchers from a wide collection of fields such as engineering, physics, cognitive science, medicine, statistics and economics have been making important contributions to the understanding, development and application of artificial systems that models certain aspects of the form and functionality of human intelligence.

An *artificial* neural network is a *model* that emulates a biological neural network. Although an artificial neuron is analogous to the biological neuron, the artificial neural networks are still far from anything close to a realistic description of how brains actually work. They nevertheless provide a rich, powerful and interesting modelling framework with proven and potential applications across sciences. Examples of such applications include Elman (1990) for learning and representing temporal structure in linguistics; Jordan (1990) for controlling and learning smooth robot movements; Gençay and Dechert (1992), Gençay (1996) and Dechert and Gençay (1996) to decode deterministic and noisy chaos and Lyapunov exponent estimations and Kuan and Liu (1995) for exchange rate prediction. Successes in these and other areas of sciences may serve as a useful addition to the tools available to the nonlinear time series modelling and prediction. In this section, we review two types of network structures, namely feedforward and recurrent networks. The structure and the learning algorithms of these networks are reviewed first. Second, we review recent theoretical contributions on the universal approximations of neural networks. Finally, we provide an explanation of why recurrent networks may be preferred over feedforward networks when they are used as noise filters.

2.1. Feedforward Networks

In a simple neural network model, the signal from input units is directly connected with the output units through the output function. The earliest form for the output function is a

threshold function, which takes a value of 0 or 1 determined by a threshold parameter. The output unit is activated when the function value is 1 and inactivated otherwise. Conventionally, this output function is called the activation function. A rich class of networks contains intermediate layers between inputs and outputs. These intermediate layers are usually called the hidden layers. A common feedforward network model with hidden layers is the single hidden layer feedforward network model. Given inputs $x_t = (x_{1,t}, \ldots, x_{n,t})$, an output of a single layer feedforward network with q hidden units is written as

$$\begin{aligned} h_{i,t} &= \Psi\Big(\gamma_{i0} + \sum_{j=1}^{n} \gamma_{ij} x_{j,t}\Big), \qquad i = 1, \cdots, q, \\ o_t &= \Phi\Big(\beta_0 + \sum_{i=1}^{q} \beta_i h_{i,t}\Big), \end{aligned} \tag{3}$$

or

$$\begin{aligned} o_t &= \Phi\Big(\beta_0 + \sum_{i=1}^{q} \beta_i \Psi(\gamma_{i0} + \sum_{j=1}^{n} \gamma_{ij} x_{j,t})\Big) \\ &=: f(x_t, \theta), \end{aligned} \tag{4}$$

where $\theta = (\beta_0, \ldots, \beta_q, \gamma_1, \ldots, \gamma_q)'$ $(\gamma_j = (\gamma_{j,0}, \ldots, \gamma_{j,n}))$ are the parameters to be estimated and Ψ and Φ are known activation functions. o_t is the estimator for the target variable y_t; x_t is the vector of inputs and h_t represents the vector of hidden units. As shown in Figure 1, the hidden units of the feedforward networks are not dynamic as they do not depend on past values generated from the networks. For this reason, the network is called a feedforward network.

Given the network structure as in equation (3) and the chosen functional forms for Ψ and Φ, a major empirical issue in the neural networks is to estimate the unknown parameters θ with a sample of data values of targets and inputs. Cognitive scientists use the following learning algorithm:

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \eta \, \nabla f(x_t, \hat{\theta}_t)[y_t - f(x_t, \hat{\theta}_t)],$$

where $\nabla f(x, \theta)$ is the (column) gradient vector of f with respect to θ and η is a learning rate. Here, $\nabla f(x, \theta)[y - f(x, \theta)]$ is the vector of the first-order derivatives of the squared-error loss: $[y - f(x, \theta)]^2$. This estimation procedure is characterized by the recursive updating or the learning of estimated parameters. This algorithm is called the method of *backpropagation*. By imposing appropriate conditions on the learning rate and functional forms of Ψ and Φ, White (1989) derives the statistical properties for this estimator. He shows that the backpropagation estimator asymptotically converges to the estimator which locally minimizes the expected squared error loss. Let y be the target variable, x be the vector for input

variables, and $f(x,\theta)$ be the network structure. The estimator θ^* to minimize the expected squared error loss is the solution of minimizing

$$E\,|y - f(x,\theta)|^2 = E\,|y - E(y|x)|^2 + E\,|E(y|x) - f(x,\theta)|^2.$$

This is equivalent to minimizing

$$E\,|E(y|x) - f(x,\theta)|^2.$$

A modified version of the backpropagation is the inclusion of the Newton direction in recursively updating $\hat{\theta}_t$ (Kuan and White, 1994). The form of this recursive Newton algorithm is

$$\begin{aligned} \hat{\theta}_{t+1} &= \hat{\theta}_t + \eta_t \hat{G}_t^{-1} \nabla f(x_t, \hat{\theta}_t)[y_t - f(x_t, \hat{\theta}_t)], \\ \hat{G}_{t+1} &= \hat{G}_t + \eta_t [\nabla f(x_t, \hat{\theta}_t) \nabla f(x_t, \hat{\theta}_t)' - \hat{G}_t], \end{aligned} \tag{5}$$

where $\hat{G}_t$ is an estimated, approximate Newton direction matrix and $\{\eta_t\}$ is a sequence of learning rates of order $1/t$. The inclusion of Newton direction induces the recursively updating of $\hat{G}_t$, which is obtained by considering the outer product of $\nabla f(x_t, \hat{\theta}_t)$. In practice, an algebraically equivalent form of this algorithm can be employed to avoid matrix inversion.

These recursive estimation (or on-line) techniques are important for large samples and real time applications since they allow for adaptive learning or on-line signal processing. However, recursive estimation techniques do not fully utilize the information in the data sample. White (1989) further shows that the recursive estimator is not as efficient as the nonlinear least squares (NLS) estimator. The NLS estimator is derived by minimizing

$$L(\theta) = \sum_{t=1}^{n} (y_t - f(x_t, \theta_t))^2. \tag{6}$$

This is a straightforward multivariate minimization problem. Conjugant gradient routines studied in Gençay and Dechert (1992) work very well for this problem.

2.2. Recurrent Networks

Applications in forecasting, signal processing and control require explicit treatment of dynamics. Feedforward networks can accommodate dynamics by including past input and target values in an augmented set of inputs. However, this kind of dynamic representation does not exploit a known feature of biological networks, that of internal feedback. Returning to a relatively simple single hidden layer network, such feedbacks can be represented as in Figure 2. In Figure 2, the hidden layer output feeds back into the hidden layer with a time

delay, as proposed by Elman (1990). The output function of the Elman network can thus be represented as

$$\begin{aligned} h_{i,t} &= \Psi\Big(\gamma_{i0} + \sum_{j=1}^{n} \gamma_{ij} x_{j,t} + \sum_{\ell=1}^{q} \delta_{i\ell} h_{\ell,t-1}\Big) \\ &=: \psi_i(x_t, h_{t-1}, \theta), \qquad i = 1, \cdots, q, \\ o_t &= \Phi\Big(\beta_0 + \sum_{i=1}^{q} \beta_i \psi_i(x_t, h_{t-1}, \theta)\Big) \\ &=: \phi_q(x_t, h_{t-1}, \theta). \end{aligned} \tag{7}$$

By recursive substitution,

$$h_{i,t} = \psi_i(x_t, \psi(x_{t-1}, h_{t-2}, \theta), \theta) = \cdots, \qquad i = 1, \cdots, q, \tag{8}$$

$$o_t = \phi_q(x_t, h_{t-1}, \theta) =: g(x^t, \theta), \tag{9}$$

where $x^t = (x_t, x_{t-1}, \cdots, x_1)$. As a consequence of this feedback, network output depends on the initial values of $h_{i,0}, i = 1, \cdots, q$ and the entire history of the system inputs, $x^t = (x_t, x_{t-1}, \cdots, x_1)$. These networks are capable of rich dynamic behavior, exhibiting memory and context sensitivity. Because of the presence of internal feedbacks, these networks are referred to as recurrent networks in the artificial neural networks literature.

The parameters of interest are θ^* which are found by minimizing $E\,|y_t - g(x^t, \theta)|^2$. Hence, $g(x^t, \theta^*)$ can be viewed as an approximation of $E(y_t|x^t)$. The network output o depends on θ directly and indirectly through the presence of lagged hidden-unit activations. Owing to this *state dependent* structure, the method of nonlinear least squares becomes infeasible. θ^* can be estimated by the recurrent backpropagation algorithm of Kuan, Hornik and White (1994) and the recurrent Newton algorithm by Kuan (1994). These algorithms are strongly consistent, provided that recurrent connection δ's are constrained suitably. In this paper, the recurrent Newton algorithm is used. It has the form:

$$\begin{aligned} \hat{e}_t &= y_t - \phi(x_t, \hat{h}_{t-1}, \hat{\theta}_t), \\ \nabla \hat{e}_t &= -\phi_\theta(x_t, \hat{h}_{t-1}, \hat{\theta}_t) - \hat{\Delta}_t\, \phi_h(x_t, \hat{h}_{t-1}, \hat{\theta}_t), \\ \hat{\theta}_{t+1} &= \hat{\theta}_t - \eta_t \hat{G}_t^{-1} \nabla \hat{e}_t\, \hat{e}_t, \\ \hat{G}_{t+1} &= \hat{G}_t + \eta_t\, (\nabla \hat{e}_t \nabla \hat{e}_t' - \hat{G}_t), \end{aligned} \tag{10}$$

where ϕ_θ and ϕ_h are column vectors of the first order derivatives of ϕ with respect to θ and h, respectively. The i-th $(i = 1, \ldots, q)$ hidden-unit activation is updated according to

$$\hat{h}_{i,t} = \Psi\Big(\hat{\gamma}_{i0,t} + \sum_{j=1}^{n} \hat{\gamma}_{ij,t} x_{j,t} + \sum_{\ell=1}^{q} \hat{\delta}_{i\ell,t} \hat{h}_{\ell,t-1}\Big) = \psi_i(x_t, \hat{h}_{t-1}, \hat{\theta}_t), \tag{11}$$

and the j-th $(j = 1, \ldots, q)$ column of $\hat{\Delta}_{t+1}$ is updated according to

$$\hat{\Delta}_{j,t+1} = \psi_{j,\theta}(x_t, \hat{h}_{t-1}, \hat{\theta}_t) + \hat{\Delta}_t\, \psi_{j,h}(x_t, \hat{h}_{t-1}, \hat{\theta}_t). \tag{12}$$

2.3. Universal Approximation

In the earlier literature, the statistical properties of the backpropagation estimator were unknown and the universal approximation was questionable. For example, there was no guidance in terms of how to choose the number of neurons and their configurations in a given layer and how to decide the number of hidden layers in a given network.

In recent years, a number of these issues have been addressed. The results in Cybenko (1989), Funahashi (1989), Hornik, Stinchcombe and White (1989, 1990), and Hornik (1991) indicate that feedforward networks with sufficiently many hidden units and properly adjusted parameters can approximate an arbitrary function arbitrarily well in useful spaces of functions with arbitrary squashing functions (e.g., logistic, hyperbolic tangent). Hornik (1991) gives the conditions under which networks with as few as a single hidden layer and arbitrary bounded and nonconstant activation functions are universal approximators. Broomhead and Lowe (1988) consider this approximation as an extension of multivariable functional interpolation.

For studying deterministic processes, such as chaos, the derivatives of an unknown function under study are also needed. A typical measure of a chaotic system is the existence of a positive largest Lyapunov exponent. Numerical measures of Lyapunov exponents require the estimation of the derivatives of the true data process (Gençay and Dechert (1992)). A poor approximation of the derivatives implies an inaccurate measure of the Lyapunov exponents. Hornik, Stinchcombe and White (1990), and Hornik (1991) show that single hidden layer feedforward neural networks can accurately approximate the derivatives of an arbitrary function. The results of the universal approximations both to the functions and its derivatives provide a suitable platform to use the feedforward network models to the approximation of a chaotic system and its Lyapunov exponents.

As indicated in the introduction, there are other methods for modelling and predicting nonlinear time series other than neural network models. For instance, the idea behind the Taylor series expansion is to increase the order of the expansion to the point where a curved surface of that order can follow the curvature of the local data points closely. The trade off with this method is that the number of terms in a multidimensional Taylor series expansion increases quite rapidly with the order. Indeed, the number of parameters needed for a Taylor series of a given order grows multiplicatively as the order of the expansion is increased and this method involves a choice of an optimal order of expansion. Casdagli (1989) points out that there are no known order of convergence results for $n > 1$, and that polynomials of high degree have an undesirable tendency to oscillate wildly.

In radial basis functions the contribution of each data point is computed by least squares and these functions are easy to implement numerically. If standard algorithms for the solution of linear systems of equations are used, Casdagli (1989) indicates that for large data sets, their implementation is no longer feasible on standard workstations. The advantages of these network models over other methods mentioned above are that feedforward network

models use only linearly *many* parameters $O(qn)$, whereas traditional polynomial, spline, and trigonometric expansions use exponentially *many* parameters $O(q^n)$ to achieve the same approximation rate (Barron (1991)). A recent survey of this literature is presented in Kuan and White (1994).

2.4. Advantages of Recurrent Networks as Noise Filters

Recurrent networks, with the consideration of the internal feedback, are more general models than the feedforward networks. The salient property of the recurrent network architecture is that the hidden unit activation functions (internal states) are fed back at every time step to provide an *additional input*. This recurrence gives the network dynamical properties which make it possible for the network to possess internal memory. Exactly *how* this internal memory is represented is not determined in advance. Instead, the network must discover the underlying temporal structure of the task and learn to encode that structure internally. In a typical feedforward network, hidden units develop representations which enable the network to perform the assigned task. The similarity structure of the internal representations reflects the demands of the task being learned, rather than the similarity of the form of the inputs. When the recurrence is added, the hidden units assume an additional function. They now provide the network with memory.

There are two important considerations as to why recurrent networks are attractive modelling tools for prediction in noisy environments. In a recurrent network architecture, the hidden unit activation functions (internal states) are fed back at every time step to provide an *additional input*. Since the recurrent network learning algorithms are sequential, the recurrence of hidden units enables the filtered data of the previous period to be used as an additional input in the current period. In other words, each time period network is subject to not only the new noisy data but the past history of all noisy inputs as well as their filtered counterparts. This additional information of filtered input history acts as an additional guidance to evaluate the current noisy input and its signal component. In contrast, filtered history never enters into the learning algorithm in a feedforward network. This is where recurrent networks differ from a feedforward network.

Secondly, because recurrent networks have the ability to keep past history of the filtered inputs as an additional information in the memory, a recurrent network would have the ability to filter noise even when the noise distribution is allowed to vary over time. In a feedforward network, however, a complete new training has to be performed with a new set of examples containing the new type of noise structure.

In a recent paper, Broomhead, Huke and Potts (1996) propose a method of noise cancellation where the *noise* is *deterministic* generated by some possibly chaotic system. Their method exploits the fact that a time series measured from a nonlinear dynamical system can be used to reconstruct the phase space of the system, and that this is still true if the time series is first passed through a linear finite impulse response filter. This paper's contribution

is to allow both deterministic or stochastic noise in the construction of nonlinear inverses from data. The recurrent network modelling provides a suitable platform for this purpose as indicated above.

3. Simulation Design

Although the universal approximation for the feedforward networks have theoretically been proved and empirically been applied to the chaotic systems (such as Lapedes and Farber (1987b), Broomhead and Lowe (1988), White (1989), Gallant and White (1992)), there are very few similar studies for recurrent networks. Since feedforward networks can be considered as the recurrent networks without feedback, the feedforward network model is a special case of the recurrent networks model. It is reasonable to assume that the universal approximation property can also be applied to the recurrent networks. In this paper, we empirically show the approximation properties for the recurrent networks using simulated data generated from well-known nonlinear deterministic systems.

3.1. Models and Data Generation

We study three well-known chaotic systems which are the Logistic map, the Hénon map and a discretized version of the Mackey-Glass delay equation. The logistic map is a one dimensional, discrete time, unimodal map

$$x_t = \beta x_{t-1}(1 - x_{t-1}). \tag{13}$$

For $\beta \in [0, 4]$ the state of the system maps itself onto itself in the closed interval $[0, 1]$. In the interval $\beta \in (3.5699, 4]$, the logistic map exhibits deterministic chaos and contains the presence of periodic as well as aperiodic cycles. Here, we set $\beta = 4$. The Hénon map is expressed by

$$\begin{aligned} x_{t+1} &= 1 - 1.4x_t^2 + z_t \\ z_{t+1} &= 0.3x_t. \end{aligned} \tag{14}$$

The matrix of derivatives of the Hénon map

$$\begin{bmatrix} -2.8\ x_t & 1 \\ 0.3 & 0 \end{bmatrix} \tag{15}$$

has a constant determinant so that the Lyapunov exponents for this map satisfy

$$\lambda_1 + \lambda_2 = ln(0.3) \approx -1.2. \tag{16}$$

The largest Lyapunov exponent of the Hénon map is 0.408. A discretized variant of the Mackey-Glass delay equation is

$$x_t = x_{t-1} + [\frac{ax_{t-s}}{1 + (x_{t-s})^c} - bx_{t-1}], \tag{17}$$

where we use $a = 0.2$, $b = 0.1$, $c = 10.0$ and $s = 17$. This equation is chosen to show the performance of the feedforward and recurrent networks with higher dimensional systems and the resulting largest Lyapunov exponent estimate. The largest Lyapunov exponent of the Mackey-Glass delay equation is 0.0086.

Our simulation study is performed both on the deterministic systems without any external noise and with additive noise. For the series without noise, we let $y_t = x_t$ and treat y_t as the target variable. The series with added noise is computed by

$$y_t = x_t + v_t$$

where v_t is the noise component. The noise is generated from the normal distribution such that $v_t = \alpha\sigma u_t$, where $\alpha = 0.01$, 0.1, 0.25 and 0.5, σ is the sample standard deviation of the x_t, and u_t is a standard normal random variable.

For each simulation model, we generate small samples with 200 observations for the logistic and the Hénon map, and large samples with 2000 observations for Mackey-Glass delay equation. The data set with 200 observations is included to measure the sensitivity of the results to small sample bias and variation. In the simulation with 200 observations, the last 20 observations are kept for the out-of-sample forecasts. For 2000 observations, the last 200 observations are reserved for forecast purposes. Let the sample size be denoted by $\{y_1, y_2, \cdots, y_n, \cdots, y_m\}$ such that n observations are utilized in the in-sample observations and the last $m - n$ are kept for the out-of-sample forecast comparisons.

The most important part of the forecast comparisons of this paper is that the results do not rely on the one time estimation of each studied model. Rather, each model is replicated 1000 times by generating a new set of data for each replication. To generate a set of sample data in each replication, a set of suitable initial values for $y_0,\ y_{-1}, \cdots, y_{-s}$ is randomly selected from a uniform distribution, then the transients are discarded before collecting observations for each replication.

3.2. Estimation and Out-of-Sample Forecasts

We let activation functions Ψ be an identity function and Φ a logistic function as in Kuan and White (1994). Each network is estimated with 4 to 6 hidden units.[1]

To estimate the unknown parameters θ in equations (3) and (7), we use both recursive Newton algorithm and nonlinear least squares (NLS) estimation. The recursive Newton

[1]This range is purely for computational considerations.

algorithms for feedforward networks and recurrent networks are described in (5) and (10), respectively. For NLS estimation, we use modified Levenberg-Marquardt algorithm. Although recursive estimator is not as efficient as the NLS estimator, the estimates from the recursive algorithm can be served as the starting values for NLS estimation (White 1989). These two methods form our two-step estimation procedure.

In the first step, 50 sets of parameters are randomly generated and the one with the lowest mean square error is chosen as the initial values for recursive estimation. We let the recursive algorithm run through the data set 10 times and use the resulting estimates as the starting values for the NLS estimation, which constitutes the second step of our estimation procedure. For the feedforward network, the final recursive estimates from the first step are used as initial values for the NLS estimation. For the recurrent network, the method of nonlinear least squares is not infeasible because of state dependence of parameters. We work around this problem by fixing the recurrent connection δ's at the final recursive estimates. The parameter β's and γ's are then estimated by the NLS method using the final estimates of the β's and γ's from the first step as initial values. Since we find that the in-sample estimation and the out-of-sample forecast from the NLS estimation are uniformly superior to those from the recursive estimation, we only report the results from the NLS.

In addition to the feedforward and recurrent network models, we also apply the ordinary least squares (LS) model to the data for comparison. For the out-of-sample forecast, we compute the one-step forecast for all models. Let $\hat{y}_{t+1}$ denote the one-step forecast value of y_{t+1}, then

$$\hat{y}_{t+1} = f(y_1, y_2, \cdots, y_t; \hat{\theta}), t = n, n+1, \cdots, m-1,$$

where $\hat{\theta}$ contains the parameters estimated from the in-sample data, $(y_1, y_2, \cdots, y_n)$. The out-of-sample comparison is then based on the one-step forecast error, $y_t - \hat{y}_t$.

3.3. Comparison Statistics

For each set of estimations, we apply several statistics to compare their in-sample fit and out-of-sample forecast performances. As a model selection criterion, we use the Schwarz Information Criterion (SIC). The SIC is computed by (Schwarz (1978))

$$SIC = \ln(\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2) + \frac{w}{n}\ln(n),$$

where w is the number of parameters in the model. The SIC is used to determine the best model among linear and neural network models. The model with the smallest SIC is the preferred model. The second term in SIC indicates that the simple estimation model with fewer number of parameters is better if both models give the same mean squared errors $(\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2)$. When two models have the same number of parameters, the comparison of SIC is the same as the comparison of the mean squared errors. We also use SIC to

determine the proper number of hidden units for neural network models.[2] Only the model with the best SIC is reported.

For the in-sample comparison, we measure errors resulting from the estimated function as well as its estimated derivatives. The squared root of the mean squared errors is defined as

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2},$$

where $\hat{y}_t$ is the in-sample estimate of y_t for time t.

To measure the errors from approximating the derivatives, we compute the difference between the true and the estimated derivatives of the studied examples. The statistic used is the squared root of the mean squared derivative errors,

$$RMSDE_{x_i} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(\frac{dy_t}{dx_{it}} - \frac{d\hat{y}_t}{dx_{it}})^2},$$

where x_i is the explanatory variable used in the estimation. Since the functions under study in our simulation are low dimensional chaotic systems, the largest Lyapunov exponent for these systems is positive. We use the Gençay and Dechert (1992) methodology to measure Lyapunov exponents. In their methodology, the estimated derivative matrices are used to calculate the Lyapunov exponents. The quality of the measure of the Lyapunov exponents is then related to RMSDE.

For the out-of-sample forecast, we use the squared root of the mean squared one-step prediction errors (RMSPE) to compare the performances of different estimation models. The computation of RMSPE is

$$RMSPE = \sqrt{\frac{1}{m-n}\sum_{t=n+1}^{m}(y_t - \hat{y}_t)^2},$$

where $\hat{y}_t$ is the one-step forecast of y_t. We also compute the ratio of the RMSPE from the neural network model to the RMSPE from the linear regression model. This ratio directly compares the forecast performance from the neural network models with that of the linear regression model. When this ratio is close to zero, the performance of the neural network model is superior to the performance of the ordinary least squares regression.

Further comparisons can be done by examining the RMSPE from the raw data. We compute the ratio of the RMSPE from the estimated model to the RMSPE from the raw data. This ratio evaluates the forecast performance for each estimated model based on the

[2]How to determine the proper number of hidden units for a neural network model is still an unsolved issue. White (1990) suggests to use the cross-validated average squared error. Here, we follow Kuan and Liu (1995) to use SIC.

prior knowledge of the simulated model. For the simulated models without noise, RMSPE from the raw data is calculated by

$$RMSPE_{DGP} = \sqrt{\frac{1}{m-n}\sum_{t=n+1}^{m}(y_t - \bar{y})^2},$$

where $\bar{y}$ is the sample mean of y_t in the out-of-sample period. In other words, $RMSPE_{DGP}$ is the sample standard deviation in the out-of-sample period. If the estimated model gives a perfect forecast, the ratio of RMSPE to $RMSPE_{DGP}$ should be very close to zero. On the other extreme, it is close to one for the worst forecast.

When the simulated model contains an additive noise, $RMSPE_{DGP}$ is defined as

$$RMSPE_{DGP} = \sqrt{\frac{1}{m-n}\sum_{t=n+1}^{m}(v_t - \bar{v})^2},$$

where $\bar{v}$ is the sample mean of v_t in the out-of-sample period. Since v_t is generated by $v_t = \alpha\sigma u_t$, the value of $RMSPE_{DGP}$ is close to the standard deviation of v_t, which is equal to $\alpha\sigma$. In the case of a perfect forecast, the forecast error will be close to the additive noise. Hence, the RMSPE from the estimated model is equal to $RMSPE_{DGP}$ and the ratio of RMSPE to $RMSPE_{DGP}$ is close to one. In the worst case, the value of RMSPE from the estimated model is close to the sample standard deviation of y_t in the out-of-sample period. This standard deviation is close to $(1+\alpha)\sigma$. Therefore, the maximum ratio of RMSPE to $RMSPE_{DGP}$ is close to $\frac{1+\alpha}{\alpha}$. For $\alpha = 0.01, 0.1, 0.25$, and 0.5, these maximum ratios are 101, 11, 5, and 3, respectively.

4. Numerical Results

Our simulation results are summarized in Tables 1 to 4. In each table, we present the in-sample and out-of-sample statistics for three estimation models. The results for each model are labeled by LS (for linear regression), RN (for recurrent networks), and FN (for feedforward networks). The number in the parenthesis after RN and FN is the selected number of hidden units based on the best SIC. The in-sample statistics in the tables are the squared root of the mean squared errors (RMSE), Schwarz Information Criterion (SIC), and the squared root of the mean squared derivative errors (RMSDE). A preferred model should deliver the smallest statistic among three models. We also show the measure of the largest Lyapunov exponent, λ_1, in the tables. Since the simulated models are chaotic, we expect a positive measure of λ_1. The accuracy of this measure is closely related to RMSDE. For the out-of-sample statistics, each table lists the squared root of mean squared prediction errors (RMSPE). The ratio of the RMSPE of the estimated model to $RMSPE_{DGP}$ is labeled

by R/DGP and presented in the second last column in each table. This ratio shows the relative forecast performance for each model. The ratio of RMSPE from the neural networks to RMSPE from the linear regression model is labeled by R/LS and presented in the last column of each table.

In each table, the first panel is the result applied to the original series without added noise. We also refer to the case without added noise as $\alpha = 0.0$. The rest of the panels contain the results for those series with added noise of $\alpha = 0.01, 0.1, 0.25$, and 0.5. In all tables, there are two values in each cell. The numbers without parentheses are the sample averages of those in-sample and out-of-sample statistics obtained from 1000 replications and the numbers within parentheses are their standard deviations.

Table 1 shows the results for the logistic map with 200 observations. Since there is only one lagged dependent variable in the true model, the first-order lag is used as an explanatory variable in the linear and the neural network models. The feedforward neural network model with six hidden units, FN(6), gives the best in-sample fit and the lowest SIC when there is no added noise. It also has the lowest out-of-sample forecast errors. The RMSPE is 0.0009 while (R/DGP) and the (R/LS) ratios are 0.0026 and 0.0025, respectively. This implies that the feedforward network model approximates the data closely both for the in-sample and out-of-sample much better than the linear model. The estimation from FN(6) also provides an accurate mean largest Lyapunov exponent estimate of 0.6879. The RMSDE is 0.0023 and is close to zero revealing that the calculated derivative vector approximates the true one closely. The standard deviations of these estimates are given within parentheses below the corresponding statistics. These standard deviations are quite small indicating the minimal sample variation of the calculated statistics over 1000 replications.

When the noise is included with $\alpha = 0.01$, the feedforward network is still the best model both for the in-sample estimation and out-of-sample forecast. For larger noise levels of $\alpha = 0.25$ and 0.5, the (R/LS) ratio drops from 3 percent to 84 percent indicating the deteriorating performance of the feedforward network in noisy environments. This deterioration is clearly noticeable in the averages of RMSPE and the RMSDE across various noise levels, as well as in the larger standard deviations of these statistics. Up to $\alpha = 0.25$, the Lyapunov exponent remains positive after which it becomes negative for $\alpha = 0.5$. The performance of the recurrent network is better than the feedforward network for $\alpha = 0.25$ and close to that of the feedforward network for $\alpha = 0.5$ in average RMSPE comparisons. Note that all (R/DGP) ratios for the linear model are fairly close to their maximum values. This implies that the linear model provides almost no forecast capability.

In Table 2, the results for the Hénon map are presented with the first lag of the dependent variable used as the explanatory variable. The reason for studying the Hénon map with one lag is to demonstrate the ability of the recurrent networks to pick up the missing second lag through its recurrent memory structure. With only one lag included in the estimation, the recurrent networks provide the best result for all out-of-sample forecast horizons ($\alpha = 0.00, 0.01, 0.1, 0.25$ and $\alpha = 0.5$). When there is no noise, the in-sample fitted errors and

out-of-sample forecast errors from recurrent networks are about 5 times lower than those from feedforward networks. This shows that recurrent neural networks can pick up the dynamic structure of the data through its recurrent memory structure. More importantly, the recurrent networks filter the data successfully and lead to more accurate out-of-sample forecasts relative to that of the feedforward model. When $\alpha = 0.01$, the recurrent network is still significantly better than the feedforward network (about 4 times better in terms of errors). This superior performance of the recurrent network to the feedforward network also declines as the values of noise increase. For up to $\alpha = 0.1$, both network models provide positive largest Lyapunov exponent estimates. At $\alpha = 0.25$ and 0.5 level, RMSDE's deteriorate significantly which lead to the negative largest Lyapunov exponent estimates.

Table 3 shows the results for the Hénon map when the first two lags are used in the estimation. Although the feedforward network model provides the best performance for both the in-sample fit and the out-of-sample forecast horizon at lower levels of noise, recurrent networks provide more accurate out-of-sample predictions at higher levels of noise ($\alpha = 0.25$ and 0.50). Both Tables 2 and 3 show that, in all cases with added noise, the ratios (R/DGP) for the linear regression model are all close to the maximum values. The linear regression model cannot capture any nonlinear dynamic for the Hénon map.

The results with the Mackey-Glass delay equation are presented in Table 4. The Mackey-Glass delay equation simulations are done with 2000 observations and the last 200 observations are kept for the out-of-sample prediction calculations. In the calculation of this equation, the first and the seventeenth lags are used as inputs in both network models. At $\alpha = 0.00$, the linear regression model provides some prediction power since its R/DGP (=0.1124) is significantly smaller than 1. Of all, the feedforward network gives the best out-of-sample predictions. The average RMSPE of the feedforward network is almost five times smaller than that of the recurrent network. The ratio R/LS for the recurrent network is as high as 0.2195. This is partly because of the prediction power observed in the linear regression model. At $\alpha = 0.01$, RMSPE is twice as much in favor of the feedforward network. Both feedforward and recurrent networks provide accurate derivative vector estimates which are reflected in the largest Lyapunov exponent estimates.

At higher levels of noise, the performance of the recurrent network dominates the performance of the feedforward network in the out-of-sample forecast comparisons. The average RMSPE comparisons rank in favor of the recurrent network models at $\alpha = 0.1, 0.25$ and 0.5. The largest Lyapunov exponent estimate is positive only for the recurrent network model at $\alpha = 0.1$. At higher levels of noise, both network models provide negative largest Lyapunov exponents. This is an indication that network models require more data to filter noise at higher levels of measurement noise.

5. Conclusions

This paper provides an informative comparison of the feedforward and recurrent network models within the framework of nonlinear signal processing methodology. An important property of the Elman type recurrent network architecture is that the hidden unit activation functions (internal states) are fed back at every time step to provide an *additional input*. This recurrence gives the network dynamical properties which make it possible for the network to possess internal memory. Exactly *how* the internal memory is represented is not determined in advance. Instead, the network must discover the underlying temporal structure of the task and learn to encode that structure internally. There are two important considerations as to why recurrent networks are attractive modelling tools for prediction in noisy environments. In a recurrent network architecture, the hidden unit activation functions (internal states) are fed back at every time step to provide an *additional input*. Since the recurrent network learning algorithms are sequential, the recurrence of hidden units enables the filtered data of the previous period to be used as an additional input in the current period. In other words, each time period network is subject to not only the new noisy data but the past history of all noisy inputs as well as their filtered counterparts. This additional information of filtered input history acts as an additional guidance to evaluate the current noisy input and its signal component. In contrast, filtered history never enters into the learning algorithm in a feedforward network. This is where recurrent networks differ from a feedforward network.

The three examples studied in this paper suggest that recurrent networks provide more accurate out-of-sample forecasts for the nonlinear prediction of noisy time series. To investigate the sources of these forecast gains, further research is needed to achieve the mathematical understanding of why and how these recurrent feedbacks improve prediction.

References

[1] Barron, A., 1991, Approximation and estimation bounds for artificial neural networks, University of Illinois at Urbana-Champaign, Department of Statistics, Technical Report 59.

[2] Broomhead, D.S., J.P. Huke, and M.A.S. Potts, 1996, Cancelling deterministic noise by Constructing Nonlinear Inverses to Linear Filters, *Physica D*, 89, 439-458.

[3] Broomhead, D.S. and David Lowe, 1988, *Complex Systems*, 2, 321-355.

[4] Casdagli, M., 1989, Nonlinear prediction of chaotic time series, *Physica D*, 35, 335-356.

[5] Cybenko, G., 1989, Approximation by superposition of a sigmoidal function, *Mathematics of Control, Signals and Systems*, 2, 303-314.

[6] Dechert, W. D. and R. Gençay, 1996, The topological invariance of Lyapunov exponents in embedded dynamics, *Physica D*, 90, 40-55.

[7] Elman, J. L., 1990, Finding structure in time, *Cognitive Science*, 14, 179-211.

[8] Farmer, J. D. and J. J. Sidorowich, 1987, Predicting chaotic time series, *Physical Review Letters*, 59, 845.

[9] Farmer, J. D. and J. J. Sidorowich, 1988, Exploiting chaos to predict the future and reduce noise, in: *Evolution, Learning and Cognition*, ed. Y. C. Lee, World Scientific, Singapore.

[10] Funahashi, K.-I., 1989, On the approximate realization of continuous mappings by neural networks, *Neural Networks*, 2, 183-192.

[11] Gallant, A. R. and H. White, 1988, There exists a neural network that does not make avoidable mistakes, *Proceedings of the Second Annual IEEE Conference on Neural Networks*, San Diego, CA, I.657-I.664, New York: IEEE Press.

[12] Gallant, A. R. and H. White, 1992, On learning the derivatives of an unknown mapping with multilayer feedforward networks, *Neural Networks*, 5, 129-138.

[13] Gençay, R., 1994, Nonlinear prediction of noisy time series with feedforward networks, *Physics Letters A*, 187, 397-403.

[14] Gençay, R., 1996, A statistical framework for testing chaotic dynamics via Lyapunov exponents, *Physica D*, 89, 261-266.

[15] Gençay, R. and W. D. Dechert, 1992, An algorithm for the n Lyapunov exponents of an n-dimensional unknown dynamical system, *Physica D*, 59, 142-157.

[16] Hornik, K., 1991, Approximation Capabilities of Multilayer Feedforward Networks, *Neural Networks*, 4, 251-257.

[17] Hornik, K., M. Stinchcombe and H. White, 1989, Multilayer feedforward networks are universal approximators, *Neural Networks*, 2, 359-366.

[18] Hornik, K., M. Stinchcombe and H. White, 1990, Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks, *Neural Networks*, 3, 551-560.

[19] Jordan, M. I., 1986, Serial Order: A parallel distributed processing approach, UC San Diego, Institute for Cognitive Science Report 8604.

[20] Kuan, C-M., 1994, A recurrent Newton algorithm and its convergence property, *IEEE Transactions on Neural Networks*, 6, 779-783.

[21] Kuan, C-M., K. Hornik and H. White, 1994, A convergence result for learning in recurrent neural networks, *Neural computation*, 6, 620-640.

[22] Kuan, C,-M. and T. Liu, 1995, Forecasting exchange rates using feedforward and recurrent neural networks, *Journal of Applied Econometrics*, 10, 347-364.

[23] Kuan, C.-M and H. White, 1994, Artificial neural networks: An econometric perspective, *Econometric Reviews*, 13, 1-91.

[24] Lapedes, A. and R. Farber, 1987a, How neural nets work, in: Neural Information Processing Systems, D. Z. Anderson, ed., AIP, New York, 442.

[25] Lapedes, A. and R. Farber, 1987b, Nonlinear signal processing using neural networks, Los Alamos National Laboratory, LA-UR-87-2662.

[26] Mañé, R., 1981, On the dimension of the compact invariant sets of certain nonlinear maps,' in *Dynamical Systems and Turbulence*, in: D. Rand and L. S. Young, eds., Lecture Notes in Mathematics 898, Springer, Berlin.

[27] Ozaki, T., 1982, The statistical analysis of perturbed limit cycle processes using nonlinear time series models, *Journal of Time Series Analysis*, 3, 29.

[28] Schwarz, G. 1978, Estimating the Dimension of a Model, *The Annals of Statistics*, 6, 461-464.

[29] Stengos, T., 1995, Nonparametric forecasts of gold rates of return, Department of Economics No: 1995-1, University of Guelph.

[30] Takens, F., 1981, Detecting strange attractors in turbulence, in: *Dynamical Systems and Turbulence*, D. Rand and L. S. Young,eds., Lecture Notes in Mathematics 898, Springer, Berlin.

[31] Tong, H., 1983, Threshold models in nonlinear time series analysis, Lecture Notes in Statistics, Vol. 21, Springer, New York.

[32] White, Halbert, 1989, Some Asymptotic Results for learning in Single Hidden-Layer Feedforward Network Models, *Journal of the American Statistical Association*, 94, 1003-1013.

[33] White, Halbert, 1990, Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings, *Neural Networks*, 3, 535-549.

[34] Yakowitz, S., 1987, Nearest-neighbor methods for time series analysis, *Journal of Time Series Analysis*, 8, 235-247.

Figure 1
The Feedforward Network

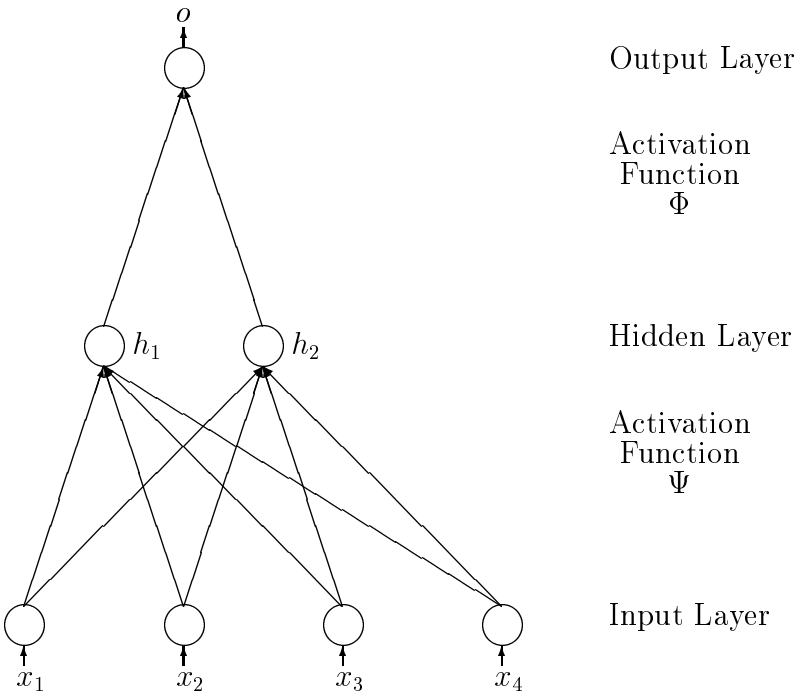

Figure 2
The Recurrent Network: The Elman (1990) Network

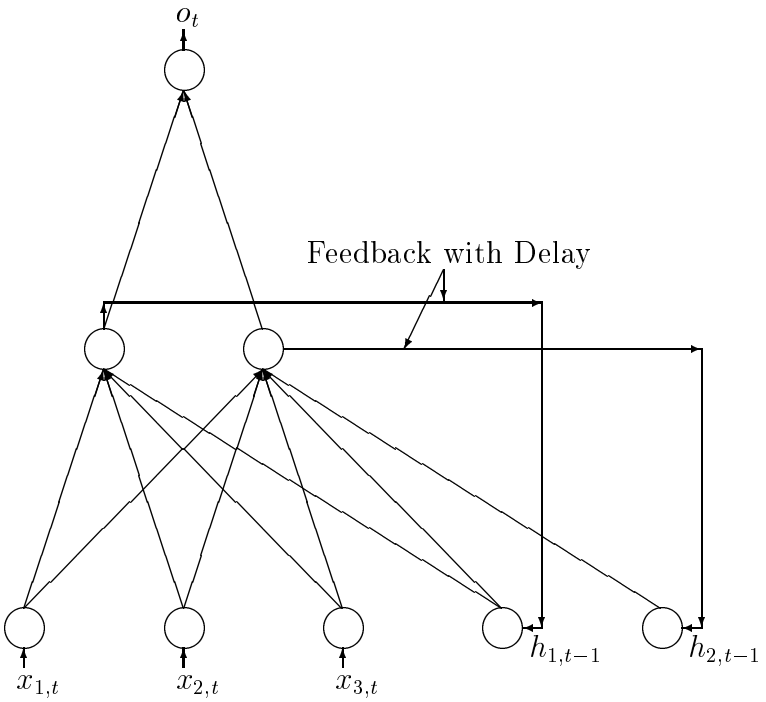

Table 1
Logistic map
200 Observations

Model	RMSE	SIC	λ_1	RMSDE	RMSPE	R/DGP	R/LS
LS	0.3515	-2.0340	-3.2668	7.9812	0.3541	1.0340	
	(0.0096)	(0.0552)	(1.1220)	(0.4182)	(0.0304)	(0.0573)	
RN(6)	0.0080	-8.8070	0.6591	0.0835	0.0077	0.0226	0.0219
	(0.0116)	(1.5430)	(0.0598)	(0.2260)	(0.0119)	(0.0346)	(0.0342)
FN(6)	0.0008	-14.2300	0.6879	0.0023	0.0009	0.0026	0.0025
	(0.0007)	(1.6320)	(0.0111)	(0.0039)	(0.0008)	(0.0023)	(0.0022)
	With Noise of $\alpha = 0.01$						
LS	0.3515	-2.0340	-3.2684	7.9821	0.3541	107.3611	
	(0.0096)	(0.0552)	(1.1258)	(0.4182)	(0.0303)	(20.7242)	
RN(5)	0.0168	-7.3140	0.6230	0.1705	0.0169	5.0338	0.0478
	(0.0208)	(0.8730)	(0.2000)	(0.4125)	(0.0204)	(5.9554)	(0.0572)
FN(4)	0.0130	-8.7060	0.6441	0.0723	0.0132	3.9412	0.0376
	(0.0291)	(0.6489)	(0.4330)	(0.6743)	(0.0289)	(8.9662)	(0.0840)
	With Noise of $\alpha = 0.1$						
LS	0.3532	-2.0240	-3.2758	8.0619	0.3559	10.7866	
	(0.0099)	(0.0568)	(1.1539)	(0.4365)	(0.0309)	(2.0717)	
RN(4)	0.0918	-3.9840	0.2939	1.8504	0.1001	2.9626	0.2827
	(0.0287)	(0.3718)	(0.5435)	(0.8890)	(0.1091)	(2.9150)	(0.3097)
FN(4)	0.0894	-4.4640	0.4273	1.6742	0.0929	2.7467	0.2626
	(0.0140)	(0.1929)	(0.2174)	(0.7082)	(0.0212)	(0.5455)	(0.0630)
	With Noise of $\alpha = 0.25$						
LS	0.3622	-1.9740	-3.2703	8.4822	0.3648	4.4173	
	(0.0116)	(0.0645)	(1.1965)	(0.5289)	(0.0347)	(0.8363)	
RN(4)	0.1900	-2.4900	0.0694	3.9617	0.1990	2.3681	0.5488
	(0.0183)	(0.1766)	(0.3809)	(1.1013)	(0.0384)	(0.3918)	(0.1086)
FN(4)	0.1936	-2.9140	0.1704	3.9888	0.2011	2.3863	0.5560
	(0.0170)	(0.1573)	(0.2903)	(1.5810)	(0.0384)	(0.3403)	(0.1076)
	With Noise of $\alpha = 0.5$						
LS	0.3927	-1.8130	-3.2286	9.9799	0.4040	2.3821	
	(0.0159)	(0.0816)	(1.1177)	(0.7990)	(0.0479)	(0.4335)	
RN(4)	0.3144	-1.4780	-0.4059	7.0742	0.3351	1.9908	0.8513
	(0.0199)	(0.1257)	(0.4918)	(1.8124)	(0.0614)	(0.3597)	(0.1555)
FN(4)	0.3189	-1.9130	-0.2845	7.3086	0.3335	1.9879	0.8471
	(0.0198)	(0.1229)	(0.4699)	(2.4974)	(0.0562)	(0.2885)	(0.1202)

Table 2

Hénon map with one lag

200 Observations

Model	RMSE	SIC	λ_1	RMSDE	RMSPE	R/DGP	R/LS
LS	0.6814	-0.7112	-1.1710	4.2388	0.6784	0.9492	
	(0.0288)	(0.0855)	(0.2333)	(0.1655)	(0.0851)	(0.0777)	
RN(4)	0.0456	-5.7230	0.3601	0.1201	0.0430	0.0603	0.0634
	(0.0356)	(1.1900)	(0.1681)	(0.1596)	(0.0388)	(0.0532)	(0.0550)
FN(4)	0.1948	-2.8960	0.3511	0.2027	0.2023	0.2837	0.3000
	(0.0072)	(0.0764)	(0.0936)	(0.2492)	(0.0247)	(0.0293)	(0.0313)
	With Noise of $\alpha = 0.01$						
LS	0.6814	-0.7111	-1.1710	4.2394	0.6784	100.8301	
	(0.0288)	(0.0856)	(0.2333)	(0.1658)	(0.0850)	(21.7886)	
RN(4)	0.0519	-5.4320	0.3582	0.1268	0.0500	7.3672	0.0744
	(0.0396)	(1.0880)	(0.1028)	(0.1943)	(0.0417)	(6.1794)	(0.0611)
FN(4)	0.1955	-2.8890	0.3500	0.2059	0.2030	30.1189	0.3007
	(0.0075)	(0.0788)	(0.1027)	(0.2763)	(0.0253)	(6.2605)	(0.0318)
	With Noise of $\alpha = 0.1$						
LS	0.6855	-0.6992	-1.1801	4.2835	0.6822	10.1359	
	(0.0293)	(0.0867)	(0.2361)	(0.1796)	(0.0853)	(2.1721)	
RN(4)	0.1607	-2.8330	0.2259	0.5130	0.1706	2.4765	0.2524
	(0.0232)	(0.2551)	(0.2053)	(0.3327)	(0.0397)	(0.5268)	(0.0617)
FN(4)	0.2470	-2.4220	0.2546	0.7067	0.2589	3.8217	0.3827
	(0.0123)	(0.1000)	(0.1104)	(0.4618)	(0.0423)	(0.7608)	(0.0626)
	With Noise of $\alpha = 0.25$						
LS	0.7067	-0.6385	-1.2317	4.5133	0.7210	4.1722	
	(0.0314)	(0.0897)	(0.2519)	(0.2376)	(0.0911)	(0.8661)	
RN(4)	0.3496	-1.2710	-0.0390	1.5860	0.3656	2.1257	0.5270
	(0.0339)	(0.1848)	(0.4127)	(0.8871)	(0.0709)	(0.3530)	(0.1071)
FN(4)	0.3988	-1.4650	0.0048	2.0427	0.4170	2.4388	0.5976
	(0.0233)	(0.1176)	(0.1875)	(2.0442)	(0.0749)	(0.4140)	(0.1114)
	With Noise of $\alpha = 0.5$						
LS	0.7766	-0.4500	-1.4044	5.3248	0.7726	2.2837	
	(0.0371)	(0.0960)	(0.3159)	(0.3885)	(0.1024)	(0.4366)	
RN(4)	0.5982	-0.1917	-0.4308	3.5706	0.6273	1.8359	0.8152
	(0.0398)	(0.1339)	(0.3805)	(1.8174)	(0.1030)	(0.2782)	(0.1183)
FN(4)	0.6285	-0.5550	-0.4455	4.2759	0.6634	1.9371	0.8599
	(0.0344)	(0.1102)	(0.5210)	(3.4277)	(0.1413)	(0.3763)	(0.1583)

Table 3
Hénon map with two lags
200 Observations

Model	RMSE	SIC	λ_1	RMSDE		RMSPE	R/DGP	R/LS
				y_{t-1}	y_{t-2}			
LS	0.6722	-0.7089	-0.5932	4.2815	0.0212	0.6683	0.9346	
	(0.0283)	(0.0852)	(0.0893)	(0.1586)	(0.0101)	(0.0848)	(0.0857)	
RN(6)	0.0140	-7.4220	0.4012	0.1618	0.3507	0.0149	0.0209	0.0225
	(0.0178)	(1.5210)	(0.0518)	(0.3505)	(1.0256)	(0.0182)	(0.0249)	(0.0280)
FN(6)	0.0025	-11.7900	0.4197	0.0044	0.0026	0.0030	0.0041	0.0044
	(0.0028)	(1.3130)	(0.0281)	(0.0125)	(0.0098)	(0.0043)	(0.0059)	(0.0063)
	With Noise of $\alpha = 0.01$							
LS	0.6722	-0.7088	-0.5934	4.2821	0.0212	0.6683	98.9642	
	(0.0283)	(0.0853)	(0.0895)	(0.1588)	(0.0102)	(0.0847)	(21.1335)	
RN(6)	0.0222	-6.0030	0.3978	0.1634	0.2777	0.0247	3.6151	0.0371
	(0.0145)	(0.6812)	(0.0523)	(0.3602)	(0.7105)	(0.0189)	(2.9005)	(0.0257)
FN(4)	0.0185	-7.5770	0.4114	0.0250	0.0222	0.0199	2.8887	0.0301
	(0.0118)	(0.4652)	(0.0350)	(0.1241)	(0.1369)	(0.0119)	(1.7133)	(0.0180)
	With Noise of $\alpha = 0.1$							
LS	0.6764	-0.6964	-0.5996	4.3257	0.0217	0.7077	9.9507	
	(0.0289)	(0.0865)	(0.0939)	(0.1719)	(0.0106)	(0.0894)	(2.1088)	
RN(4)	0.1539	-2.8220	0.2427	0.9678	0.4578	0.1675	2.4418	0.2521
	(0.0392)	(0.3570)	(0.1524)	(0.5470)	(0.4332)	(0.0501)	(0.7059)	(0.0744)
FN(4)	0.1474	-3.3410	0.3394	0.6347	0.3191	0.1618	2.3518	0.2435
	(0.0121)	(0.1615)	(0.0671)	(0.3828)	(0.3940)	(0.0333)	(0.4100)	(0.0554)
	With Noise of $\alpha = 0.25$							
LS	0.6978	-0.6342	-0.6290	4.5532	0.0237	0.6933	4.0995	
	(0.0312)	(0.0902)	(0.1117)	(0.2302)	(0.0124)	(0.0886)	(0.8428)	
RN(4)	0.3301	-1.2710	0.0367	2.0688	0.6383	0.3664	2.1400	0.5349
	(0.0436)	(0.2373)	(0.2998)	(1.5231)	(1.3944)	(0.0894)	(0.4575)	(0.1352)
FN(4)	0.3348	-1.6990	0.1910	1.7366	0.5969	0.3697	2.1433	0.5393
	(0.0237)	(0.1423)	(0.1220)	(0.8508)	(1.0157)	(0.0941)	(0.4796)	(0.1248)
	With Noise of $\alpha = 0.5$							
LS	0.7685	-0.4417	-0.7270	5.3576	0.0300	0.7639	2.2499	
	(0.0373)	(0.0976)	(0.1733)	(0.3829)	(0.0171)	(0.1026)	(0.4263)	
RN(4)	0.5722	-0.1613	-0.2751	4.1252	0.8721	0.6309	1.8498	0.8287
	(0.0421)	(0.1468)	(0.3161)	(3.1302)	(1.6834)	(0.1105)	(0.3061)	(0.1280)
FN(4)	0.5797	-0.5995	-0.1122	3.8401	0.9877	0.6402	1.8630	0.8398
	(0.0368)	(0.1273)	(0.2004)	(2.1869)	(1.8826)	(0.1475)	(0.3556)	(0.1564)

Table 4
Mackey-Glass delay equation
2000 Observations

Model	RMSE	SIC	λ_1	RMSDE		RMSPE	R/DGP	R/LS
				y_{t-1}	y_{t-17}			
LS	0.0255	-7.3280	-0.0070	0.0012	0.0485	0.0255	0.1124	
	(0.0002)	(0.0181)	(0.0005)	(0.0002)	(0.0003)	(0.0012)	(0.0047)	
RN(6)	0.0058	-10.7200	0.0635	0.0626	0.0113	0.0056	0.0247	0.2195
	(0.0071)	(1.5370)	(0.0678)	(0.1103)	(0.0140)	(0.0071)	(0.0308)	(0.2731)
FN(6)	0.0012	-13.6500	0.0075	0.0001	0.0010	0.0012	0.0052	0.0460
	(0.0006)	(1.0530)	(0.0014)	(0.0002)	(0.0008)	(0.0006)	(0.0025)	(0.0223)
	With Noise of $\alpha = 0.01$							
LS	0.0257	-7.3110	-0.0070	0.0012	0.0486	0.0258	11.396	
	(0.0002)	(0.0179)	(0.0005)	(0.0002)	(0.0003)	(0.0011)	(0.7660)	
RN(6)	0.0069	-9.9660	0.0542	0.0531	0.0111	0.0068	3.0119	0.2646
	(0.0048)	(0.9422)	(0.0607)	(0.1115)	(0.0155)	(0.0048)	(2.1429)	(0.1864)
FN(5)	0.0036	-11.1700	0.0068	0.0001	0.0009	0.0036	1.5901	0.1400
	(0.0002)	(0.1197)	(0.0015)	(0.0002)	(0.0007)	(0.0003)	(0.1207)	(0.0129)
	With Noise of $\alpha = 0.1$							
LS	0.0417	-6.3420	-0.0117	0.0008	0.0490	0.0417	1.8441	
	(0.0006)	(0.0279)	(0.0006)	(0.0001)	(0.0004)	(0.0017)	(0.0621)	
RN(4)	0.0308	-6.8500	0.0024	0.0641	0.0823	0.0307	1.3549	0.7352
	(0.0051)	(0.3545)	(0.0501)	(0.0538)	(0.1281)	(0.0055)	(0.2357)	(0.1279)
FN(4)	0.0330	-6.7540	-0.0016	0.0052	0.0106	0.0330	1.4563	0.7900
	(0.0007)	(0.0438)	(0.0023)	(0.0046)	(0.0048)	(0.0020)	(0.0546)	(0.0263)
	With Noise of $\alpha = 0.25$							
LS	0.0850	-4.9190	-0.0412	0.0001	0.0533	0.0850	1.5004	
	(0.0016)	(0.0366)	(0.0022)	(0.0001)	(0.0008)	(0.0045)	(0.0465)	
RN(4)	0.0657	-5.3750	-0.0062	0.1116	0.2327	0.0656	1.1584	0.7728
	(0.0152)	(0.5555)	(0.0688)	(0.0718)	(0.2564)	(0.0160)	(0.2763)	(0.1826)
FN(4)	0.0775	-5.0450	-0.0172	0.0648	0.0925	0.0777	1.3729	0.9150
	(0.0017)	(0.0440)	(0.0063)	(0.0139)	(0.0193)	(0.0046)	(0.0539)	(0.0229)
	With Noise of $\alpha = 0.5$							
LS	0.1550	-3.7160	-0.0502	0.0076	0.0835	0.1550	1.3688	
	(0.0027)	(0.0348)	(0.0046)	(0.0011)	(0.0031)	(0.0079)	(0.0411)	
RN(4)	0.1197	-4.1920	-0.0014	0.1452	0.3524	0.1203	1.0627	0.7764
	(0.0300)	(0.6256)	(0.0671)	(0.0724)	(0.2658)	(0.0311)	(0.2731)	(0.1977)
FN(4)	0.1430	-3.8190	-0.0257	0.1648	0.2044	0.1438	1.2703	0.9280
	(0.0029)	(0.0408)	(0.0068)	(0.0203)	(0.0255)	(0.0081)	(0.0481)	(0.0240)