Econometric Institute Rapport $EI2000-31A$

Neural networks as econometric tool

Johan F. Kaashoek (kaashoek@few.eur.nl)
Econometric Institute, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

Herman K. van Dijk (hkvdijk@few.eur.nl)
Econometric Institute and Tinbergen Institute, Erasmus University Rotterdam, P.O.Box 1738, 3000 DR Rotterdam, The Netherlands

Abstract. The flexibility of neural networks to handle complex data patterns of economic variables is well known. In this survey we present a brief introduction to a neural network and focus on two aspects of its flexibility . First, a neural network is used to recover the dynamic properties of a nonlinear system, in particular, its stability by making use of the Lyapunov exponent. Second, a two-stage network is introduced where the usual nonlinear model is combined with time transitions, which may be handled by neural networks. The connection with time-varying smooth transition models is indicated. The procedures are illustrated using three examples: a structurally unstable chaotic model, nonlinear trends in real exchange rates and a time-varying Phillips curve using US data from 1960-1997.

1. Introduction

In recent decades one witnesses a substantial increase in the interest of econometricians for nonlinear models and methods. This is due to: (i) advances in processing power of personal computers; (ii) increased research on algorithms for fast numerical optimization methods; and (iii) the availability of large data sets. One of the nonlinear models which received much attention from applied researchers is a neural network, also known as neural net. The basic idea behind a neural net is the tremendous data-processing capability of the human brain. Human brains consist of an enormous number of cells, labeled neurons. These neurons are connected and signals are transmitted from one cell to an other cell through the connections. These connections are, however, not all equally strong. When a signal is transmitted through a strong connection it arrives more strongly in the receiving neuron. One may argue that there is a particular weight associated with each connection which varies with the strength of the connection. Neurons may also receive signals from outside the brains. These are then transformed within

the brains and returned to the outside world. The whole structure of signal-processing between many (unobserved) cells can be described by a particular mathematical model which is therefore known as a neural network model. For a more detailed description of the analogy between the mathematical neural network models and the working of the human brain we refer to, e.g., Simpson (1990).
Neural networks are used in many sciences like biology, informatics, and econom(etr)ics. Within the latter field neural nets are, in particular, applied for the description and prediction of complex data patterns in economic time series. The field is very extensive and empirical illustrations are many. This paper is not intended to give a complete survey. Instead, we start with a brief introduction on neural nets and their flexibility. Our focus is on the following two applications of neural network analysis with the aim of showing that neural nets are a convenient econometric tool:

(i) Recovery of the unobserved dynamics, in particular, stability of a nonlinear system from a low dimensional data set;

(ii) Specification of a neural network where a time varying component is included.

In the first topic a neural net is used to recover the dynamic properties of a nonlinear system, in particular, its stability by making use of the Lyapunov exponent. We use one simulated series from a structurally unstable chaotic model and some data from real exchange rates to illustrate the methods. Second, a two-stage network is introduced where the usual nonlinear model is combined with time transitions which may be handled by neural nets. The connection with time- varying smooth transitions models is indicated. The procedures are illustrated on a time-varying Philips curve using US data from 1960-1997. We discuss connections with the existing literature but refer for a general introduction to neural networks to Hertz, Krogh and Palmer (1991) and Bishop (1995) and the references cited there.

2. A simple introduction to neural networks

There exist many classes of neural networks, see e.g. Hertz, Krogh and Palmer (1991) and Bishop (1995). In this paper we restrict attention to a simple class which is known as the three-layer feed forward neural network, also labeled the Rumelhart-Hinton-Williams multi-layer network after Rumelhart et al. (1986). For expository purpose we describe and interpret this network as a generalization of the well known linear model from basic econometrics, see e.g. Theil (1971), chapter 3. Suppose that the cells of the network are partitioned into particular groups or layers and suppose further that there exist three such layers: the 'input' layer, the 'hidden' layer, and the 'output layer'. The cells of the input layer correspond to the 'regressors' or 'explanatory variables' in the standard linear regression model. The cells in the output layer correspond to the dependent variables in the linear model. The hidden layer contains cells which transmit the signals from the input layer to the output layer. These cells may be interpreted as unobserved components built into the linear model. A graph of a neural network with three cells in the input layer, two cells in the hidden layer and two cells in the output layer is shown in figure 1.

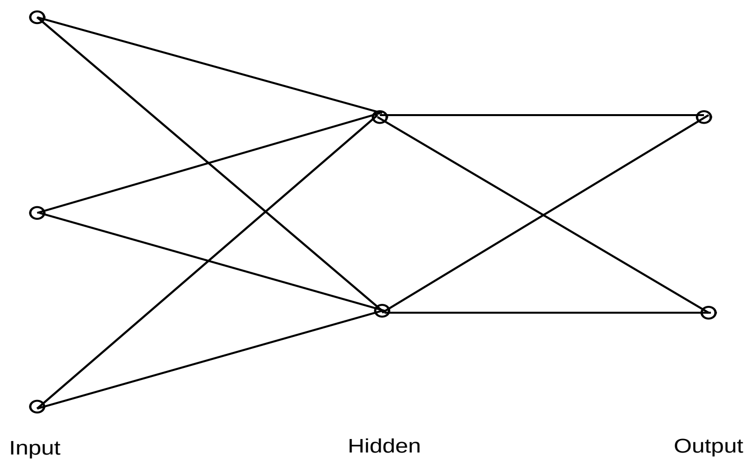

Figure 1. Graph of a neural network

The network transmit signals as follows. A weighted sum of the signals of the input cells are sent to the hidden layer cells. Within the cells of this layer the values of the signals received are transformed by a so-called 'activation function'. A weighted sum of the transformed signals is then sent to the cells of the output layer. We note that the weights in the neural network correspond to unknown parameters in the linear model.
Henceforth, we make use of the following (standard) notation. A neural network with I cells in the input layer, H cells in the Hidden Layer and O cells in the output layer is denoted as $nn(I, H, O)$. In figure 1 the network is given as $nn(3, 2, 2)$.

Next, we discuss the mathematical structure of a neural net. We make use of the following notation for cells, signals and weights:

i	index of input cells, $i = 1, \cdots, I$
h	index of hidden layer cells, $h = 1, \cdots, H$
j	index of output cells, $j = 1, \cdots, O$
$g(.)$	activation function
x_i	value of input cell i
y_j	value of output cell j

a_{ih}	weight of the signal from input cell i to hidden cell h
b_h	constant input weight for hidden cell h
c_{jh}	weight of the signal from hidden cell h to output cell j
d_j	constant weight for output cell j

The value of the signal that arrives in hidden cell h is given as

$$\text{input for hidden cell } h = \sum_{i=1}^{I} (a_{ih} x_i) + b_h \tag{1}$$

Hidden cell h transforms the value of this signal with the activation function $g(.)$ as follows

$$\text{output from hidden cell } h = g(\sum_{i=1}^{I} (a_{ih} x_i) + b_h) \tag{2}$$

where the activation function is a monotonous increasing and bounded function given as

$$g(x) = \frac{1}{1 + e^{(} - x)} \tag{3}$$

which is the well known logistic function, defined on the interval $[0, 1]$. A particular value of the logistic function indicates the extent to which a hidden cell is activated. The logistic function has attractive properties such as that the derivative is equal to $g(x)(1 - g(x))$. The graph of the function is given in figure (2) as

Other choices of the activation function are other monotone squashing functions as the *arctan* and *tanh* functions and the *cosine* squashing function, see e.q. Hertz, Krogh and Palmer (1991).

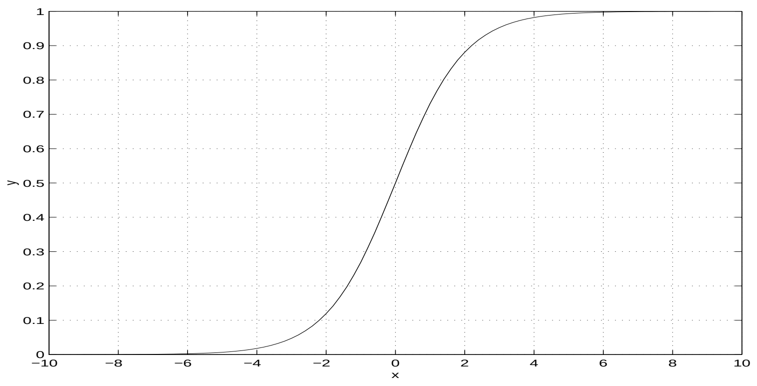

Figure 2. Graph of logistic activation function

Next, the value y_j of the output cell j is given as the weighted sum of the output of the hidden cells. It is equal to

$$y_j = \sum_{h=1}^{H} c_{jh}\, g\left(\sum_{i=1}^{I} a_{ih} x_i + b_h\right) + d_j \tag{4}$$

In matrix notation one can write:

$$y = C\mathcal{H} + d \tag{5}$$
$$\mathcal{H} = G(xA + b) \tag{6}$$

where

$x \in \mathbb{R}^I$	$y \in \mathbb{R}^O$
$A = (a_{ih}), I \times H$ matrix	$b \in \mathbb{R}^H$
$\mathcal{H} = (h_1, \cdots, h_H$, the vector of hidden cells outputs	
$G : \mathbb{R}^H \rightarrow \mathbb{R}^H$	
is the vector function, given by	
$G(v) = [g(v_1, \cdots, g(v_H)]'$	
$C = (c_{jh}), O \times H$ matrix	$d \in \mathbb{R}^O$

The neural network $nn(.)$ describes the situation at one moment in time and it indicates a deterministic relation. By adding a subscript t and an error term ϵ one obtains the system

$$\begin{aligned} y_t &= C\mathcal{H}_t + d + \epsilon_t \\ \mathcal{H}_t &= G(x_t A + b) \end{aligned} \tag{7}$$

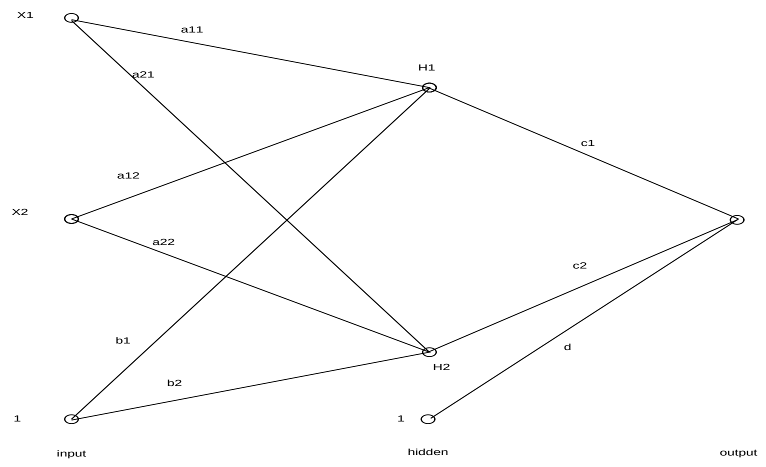

Figure 3. Graph of a neural network

Figure 3 is a representation of an $nn(3,2,1)$. The mathematical specification of this model is equal to

$$y_t = d + \frac{c_1}{1 + e^{-a_{11}y_{t-1} - a_{12}y_{t-2} - b_1}} + \frac{c_2}{1 + e^{-a_{21}y_{t-1} - a_{22}y_{t-2} - b_{12}}} \tag{8}$$

We note that this model is closely related to a threshold autoregressive model; for details see Granger and Teräsvirta (1993) and Van Dijk (1999).

2.1. Flexibility of neural networks

The flexibility of three layer feed forward neural nets is well documented. It is summarized by its so-called 'universal approximation' property. Most of this approximation theory starts with Kolmogorov's representation theorem, see (Kolmogorov, 1957). This provides the background for the Hecht-Nielsen article in 1987, see (Hecht-Nielsen, 1987). From the point of view of a neural network user, the Kolmogorov theorem provides, however, a justification for the existence of approximations in the reverse way. That is to say, the number of layers and cells are given but not the functional form of the one-dimensional activation functions. In a neural network one encounters the opposite case: the activation functions g are given (to some extent) but, at least, the number of hidden layer cells is unknown.
The articles of Gallant and White (1988) (with the revealing title: "There exists a neural network that does not make avoidable mistakes", Hecht-Nielsen (1989), Cybenko (1989), Funahashi (1989) and Hornik, Stinchcombe and White (1989) provides the theoretical background for the statement :

A (three layer) feed forward network is an universal approximator.

This general statement should be interpreted in the sense that any square integrable function can be approximated arbitrary close in L_2 norm.

Further, the articles of Hornik, Stinchcombe and White (1990) and Gallant and White (1992) extend the approximation capabilities of the network to the derivative of a function.

2.2. Estimation of parameters of neural networks

An generally accepted optimization principle is to minimize the norm of

$$\min_{\theta} \sum_{t} ||y_t - \hat{y}_t(\theta)||^2 \tag{9}$$

where $||.||$ is the Euclidean norm. For the case of a neural net it follows that one minimizes the criterion function

$$\min_{A,b,C,d} \sum_{t} ||y_t - CG(x_t A + b) - d||^2 \tag{10}$$

A well known method for numerical optimization is the simplex method and the BFGS method, a gradient method; see Press et al. (1988).

2.3. Determining the size of a neural network

Neural nets are flexible, but the price of increased flexibility is the danger of 'overfitting'. This statement may be explained as follows. In empirical econometric models one assumes that an observed economic time series consists of a part that can be explained and a part that is labeled unexplained or 'residual noise'. With 'overfitting' this noise is also 'fitted'. Then one obtains a wrong picture of the real data generating process and the quality of the forecasts may be badly affected. 'Overfitting' with neural nets may occur by increasing the number of hidden cells, which increases the number of parameters, without increasing the number of explanatory variables or inputs. Because of this possibility neural nets are more sensitive to 'overfitting' than other classes of models like autoregressive models. Therefore it is important to develop methods that determine the optimal size of a neural net. Pruning methods apply to the reduction of large neural networks to smaller ones. Two methods can be distinguished: weight (inter-connection) reduction or node reduction. Examples can be found in Hertz, Krogh and Palmer (1991) and Bishop (1995); see also Mozer and Smolensky (1989).

Below we present a brief description of a descriptive method to reduce the size of a neural net. For more details we refer to Kaashoek and van Dijk (1998).

2.3.1. *Pruning a network: the incremental contribution method*

The method we follow is labeled 'incremental contribution method". It looks for each cell separately how much the specific cell contributes to the overall performance of the network. When this contribution is considered to be low then such a cell is a candidate for excluding from the existing network (and all its connections). Re-estimating the reduced network may confirm this exclusion. To measure the contribution of a cell we look at two quantities.
First, the square of the correlation coefficient R^2 between y and $\hat{y}$, the neural network output where

$$R^2 = \frac{(\hat{y}'y)^2}{(y'y)(\hat{y}'\hat{y})} \tag{11}$$

where y as well $\hat{y}$ are taken in deviation of the means.
The procedure applies to hidden layer cells as well as to input layer cells, but here we restrict ourselves to hidden layer cells.
The contribution of cell h can now be measured by leaving out cell h, and its connection from the network, and again calculate the square of the correlation coefficient; we denote network estimates with cell h left out by $\hat{y}_{-h}$, and the corresponding r^2 is defined as:

$$R^2_{-h} = \frac{(\hat{y}'_{-h}y)^2}{(y'y)(\hat{y}'_{-h}\hat{y}_{-h})}. \tag{12}$$

The incremental contribution $R^2_{incr}(i)$ is now given as

$$R^2_{incr}(h) = R^2 - R^2_{-h}. \tag{13}$$

In the group of hidden layer cells, cells with a low R^2_{incr} are candidates for exclusion.

The second quantity involves the idea of principal components, see Theil (1971). Let again $\hat{y}_h$ be the network output with exclusion hidden layer cell h. Construct the vector e_{-h} of residuals

$$e_{-h} = y - \hat{y}_{-h} \tag{14}$$

and the matrix E_{-H}:

$$E_{-H} = (e_{-1}, \cdots, eh_{-h}). \tag{15}$$

The matrix $E_{-H}{}'E_{-H}/T$ is an estimate of the covariance matrix of the e_{-h}'s. The principal component of $E_{-H}{}'E_{-H}$ is the eigenvector at maximal eigenvalue in absolute sense. Hence the principal components provides a linear combination of $(e_{-1}, \cdots, e_{-h})$ which explains the largest part of the variance.
Which fraction is explained by each of the eigenvectors is given by the relative weight w_i with

$$w_h = \frac{\lambda_h}{\sum_{h=1}^{H} \lambda_h} \tag{16}$$

where λ_h are the eigenvalues of $E_{-H}{}'E_{-H}$. In case of the principal component λ_h is the largest eigenvalue.
Now one can look at the components of the principal component itself: cells with low incremental contribution will have a low (in absolute sense) coefficient in the eigenvector composing the principal component. The exclusion of those cells will cause a relatively small increase in the residuals; these cells are again candidate for exclusion.

Finally, one may apply a graphical analysis. Assuming the graph of $\hat{y}$ fits well with the graph of y, the graph of $\hat{y}_{-h}$ may differ less from the graph of y for those cells with a "low contribution".

The procedure involves the contributions of one cell only. It is a "feature" of neural networks that sometimes pair of cells do have a similar contribution with the output of the cells having reverse sign. Such a "behaviour" can be detected by graphical analysis and by observing that in the principal component analysis, explained above, such type of cells do have (almost) equal coefficients. In that case one has to look at the incremental contribution of both cells together.

The reduction method is applied in the next sections.

3. Stability analysis of complex nonlinear systems

A linear autoregressive system of equations of the n-vector of variables x(t) can be written as

$$x_{t+1} = Ax_t, x_t \in \mathbb{R}^N, \tag{17}$$

The stability of fixed or equilibrium points depends on the eigenvalues λ (real or complex) of the matrix A. Taking the absolute value or modulus of the eigenvalues, $\|\lambda\|$, the fixed point will be unstable if for some eigenvalue λ, $\|\lambda\| > 1$ which is equivalent with $\ln\|\lambda\| > 0$. The same holds for the stability of orbits or time series $x_0, \cdots, x_{T-1}, \cdots$. Note that in a point $x_t \in \mathbb{R}^N$, the logarithm of the local expansion rate in the direction of a vector $v \in \mathbb{R}^N$ is given as

$$\ln\|A\frac{v}{\|v\|}\| \tag{18}$$

Lyapunov exponents are a generalization of the above concept for non-linear systems. They are defined as the (spatial or time) means of the logarithm of local expansion rates. In the case of time means, the expansion rates are calculated in the time series $x_0, \cdots, x_{T-1}, \cdots$ where in a point $x_t \in \mathbb{R}^N$, the logarithm of the local expansion rate in the direction of a vector $v \in \mathbb{R}^N$ is now given as

$$\ln\|D_xF(x_t)\frac{v}{\|v\|}\| \tag{19}$$

where $F : \mathbb{R}^N \to \mathbb{R}^N$ is the data generating function:

$$x_{t+1} = F(x_t) \tag{20}$$

and D_xF is the jacobian. Then the Lyapunov exponent λ_{v_0}, with start direction vector v_0 and start value x_0, is defined as

$$\lambda_{v_0} = \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\ln\|D_xF(x_{t-1})\frac{v_{t-1}}{\|v_{t-1}\|}\| \tag{21}$$

$$v_t = D_xF(x_{t-1})v_{t-1},\ \|v_0\| = 1. \tag{22}$$

The dependence of λ on v_0 and on x_0 seems to indicate an infinite number of Lyapunov exponents. However, this is not the case. In general there are as much Lyapunov exponents as the dimension N of the system; see (Guckenheimer and Holmes, 1983). Above all, for an ergodic system, with the space-mean being equal to time-mean, it follows that for almost all start directions v_0, and for almost all start values x_0, the value of λ_{v_0} will be the largest Lyapunov exponent; see (Arnold and Avez, 1988) and (Guckenheimer and Holmes, 1983).

Since the mean of logarithms is the logarithm of the geometric mean, one can also write

$$\lambda_{v_0} = \lim_{T\to\infty} \frac{1}{T} \ln \prod_{t=1}^{T} \|D_x F(x_{t-1}) \frac{v_{t-1}}{\|v_{t-1}\|}\| \tag{23}$$

$$= \lim_{T\to\infty} \frac{1}{T} \ln \|D_x F^T(x_0) v_0\| \tag{24}$$

If some (or all) of the Lyapunov exponents are positive then one has so-called "sensitivity on start values", a characteristic of chaotic series. Hence, especially, the largest Lyapunov exponent is of interest: if positive, then the series is unstable, if negative then the series is stable.

This result can directly be applied if F, the data generating function is known. However, in practice, one observes only a one dimensional, finite time series $\{x_t, t = 0, \cdots, T-1\}$. So the question is how to extract from the series $\{x_t\}$ the dynamic properties, especially the value of the largest Lyapunov exponent of the original (unknown) model (20).
Although only the series x_t is given, one can use the embedding theorem by Takens (Takens, 1981), to reconstruct from the one-dimensional series x_t the original deterministic and smooth model. This theorem says that if the data x_t has an *deterministic explanation*, which means the data generating process is smooth and deterministic, there exists a finite embedding m, such that the dynamic system $\Psi : \mathbb{R}^m \to \mathbb{R}^m$ given by

$$\Psi : (x_t, x_{t-1}, \cdots, x_{t-m+1}) \to (x_{t+1}, x_t, \cdots, x_{t-m+2}) \tag{25}$$

has the same dynamical properties as the original one. Moreover, if the original system is N-dimensional then an embedding dimension $m = 2N + 1$ will be sufficient to have "Ψ reconstruct the original system".
Note that the only unknown component of Ψ is the first component function $\Psi_1 : \mathbb{R}^m \to \mathbb{R}$ with

$$\Psi_1 : (x_t, x_{t-1}, \cdots, x_{t-m+1}) \to x_{t+1}. \tag{26}$$

For the other components, $i > 1$, yields $\Psi_i(x_t, x_{t-1}, \cdots, x_{t-m+1}) = x_{t-i+2}$.
This function Ψ_1 should be approximated, and a neural network seems to be a proper candidate to do the job.

Once Ψ is found, the Lyapunov exponents can be calculated using equation (23). This means the calculation of $D_x\Psi$ in each point of the time series $(x_t, x_{t-1}, \cdots, x_{t-m+1})$. Note that $D_x\Psi$ has the form of a companion matrix. However, to avoid overflow in calculating the product of $D_x\Psi$'s the more stable method of Eckmann-Ruelle is used. In this case, a QR decomposition of $D_x\Psi$ is calculated at each point and the Lyapunov exponents are now simply the products of diagonal elements of the matrices R; see Eckmann and Ruelle (1985).

We apply this procedure to two data sets, one simulated and one economic time series.

3.1. Simulated data experiment

In this experiment we use simulated data. The data are generated by the model

$$\begin{aligned} x_{t+1} &= \alpha x_t + \epsilon_t - 0.5 \\ \epsilon_{t+1} &= \gamma \epsilon_t (1 - \epsilon_t), \end{aligned} \tag{27}$$

where only the series x_t is observed. The data, called $CH95$, are generated with $\alpha = 0.95$ and $\gamma = 4$. Although completely deterministic, this model has some nice features:

- The series ϵ_t is chaotic: for start values in $[0,1]$, the series ϵ_t is bounded between 0 and 1 but has the "sensitivity of start values" property characteristic for chaotic series.
- The model is structural unstable ($\gamma = 4$); e.g. a small change in the coefficient value 4, will cause a dynamical different data series. For instance, a γ value greater than 4 will for almost all start values (between 0 and 1) generate diverging (exploding) data ϵ_t; for values less than 4, periodic data are possible: the system is *structural* unstable.

Suppose only the one dimensional data set x_t is observed. Our goal should be to extract from this series, the dynamical properties of the original data generating process: only Lyapunov exponents are considered.

The original data generating model (27) has two Lyapunov exponents which can be calculated analytically. In a point (x_i, ϵ_i), the jacobian of the system function F is given by

$$\begin{pmatrix} 0.95 & 1 \\ 0 & 4(1-\epsilon_i) \end{pmatrix} \tag{28}$$

Since the jacobian $D_xF^T(x_0,\epsilon_0) = \prod_{t=1}^{T} D_xF(x_{t-1},\epsilon_{t-1})$, one has

$$D_xF^T(x_0,\epsilon_0) = \begin{pmatrix} 0.95^T & a_{12} \\ 0 & a_{22}(T) \end{pmatrix} \tag{29}$$

where $a_{22}(T) = \prod_{t=1}^{T} 4(1 - \epsilon_{t-1})$. It is obvious that $D_x F^T$ has two eigenvalues, e.a. 0.95^T and $a_{22}(T)$. With $v_0 = (1, 0)$, $D_x F^T(x_0, \epsilon_0)v_0 = ((0.95)^T, 0)$ which results in a Lyapunov exponent value of $\ln(0.95)$.
It is well known that the system $\epsilon_t = 4\epsilon_{t-1}(1 - \epsilon_{t-1})$ has Lyapunov exponent $\ln(2)$. Since $\ln(2) > \ln(0.95)$, for any start vector $v_0 \neq (1, 0)$, the resulting Lyapunov exponent will be $\ln(2)$.
Writing both Lyapunov exponents in base 2 logarithm, the values will be $\ln(0.95)/\ln(2) \sim 0.074$ and 1.

Before starting with the neural network computations of the function Ψ_1, see equation (26), first the scatter diagram $\{(x_{t-1}, x_t\}$ of the series $CH95$ is given. The sample size T is 200. Since e_t is bounded between 0 and 1, the graph of (x_{t-1}, x_t) lies between the lines $y = 0.95x + 0.5$ and $y = 0.95x - 0.5$. see figure (4).

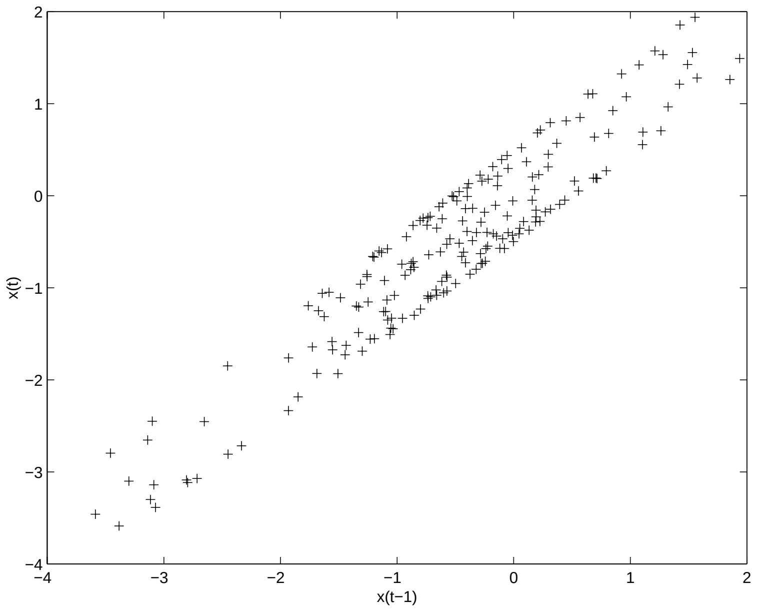

Figure 4. Scatter diagram of series $CH95$.

As said above, in order to extract from the one dimensional observed data $\{x_t\}$, the dynamic properties, especially the value of the largest Lyapunov exponent of the original model (27), only the function

$$\Psi_1(x_t, x_{t-1}, \cdots, x_{t-m+1}) = x_{t+1}$$

should be approximated by a neural network. This implies a choice for the value of m, or in neural network terms, the size of the input layer.

Since the original system has dimension 2, based on Takens embedding theorem, it will be (more than) sufficient to take as neural network input variables $(x_t, x_{t-1}, \cdots, x_{t-5})$; adding an additional constant, the dimension of the input layer will be 6.
The initial number of hidden layers will be 5, while the output layer has only one cell, the target value being x_{t+1}.

To summarize the performance of this network, the quantities

$$R^2 = \frac{(\hat{y}'y)^2}{(y'y)(\hat{y}'\hat{y})} \tag{30}$$

$$MSSR = \frac{1}{T}(y-\hat{y})'(y-\hat{y}) \tag{31}$$

$$SIC = \ln(MSSR) + \frac{n_p}{2T}\ln(T), \tag{32}$$

are calculated. In table (I) the results for the $nn(6,5,1)$ are reported.

Table I. Results of $nn(6,5,1)$

R^2	$MSSR$	SIC
1.00	$1.4\,10^{-9}$	-9.70

In table (II) the incremental contributions of the hidden layer nodes are given; both R^2 and principal component vector (row with label PrincComp in table (II)) indicate that three hidden layer nodes can be removed.

Table II. Contribution of hidden layer nodes

Hidden cell	1	2	3	4	5
R^2_{incr}	0.000	0.000	0.517	0.880	0.940
PrincComp.	−0.00	−0.000	−0.081	−0.733	0.675

The incremental contributions of inputs are shown in table (III); the inputs $x_{t-4}, x_{t-3}, x_{t-2}$ should be removed.
Applying node removal, the network is reduced to two inputs plus constant and three hidden layer nodes (plus constant). This $nn(3,3,1)$ performs as well as the larger network; see table (IV).

Both input nodes do have the same contributions; no further reduction at this level is applied.

Table III. Contribution of input layer nodes

Input cell	x_{t-4}	x_{t-3}	x_{t-2}	x_{t-1}	x_t
R^2_{incr}	0.000	0.000	0.000	0.124	0.241
PrincComp.	0.000	0.000	-0.000	0.613	0.790

Table IV. Results of $nn(3,3,1)$ and $nn(3,2,1)$

Network	R^2	$MSSR$	SIC
$nn(3,3,1)$	1.00	$1.5\,10^{-9}$	-10.04
$nn(3,3,1)$	1.00	$6.3\,10^{-9}$	-9.31

Table V. Contribution of hidden layer nodes in $nn(3,3,1)$ network

Hidden cell	1	2	3
R^2_{incr}	0.696	0.890	0.943
PrincComp.	0.114	-0.745	0.657

The incremental contributions of hidden layer nodes is shown in table (V). Based on the principal component vector (with weight 99%) hidden node 1 could be removed; see again table (IV) for the results on this network. The performance is just slightly worse compared to the larger network. No further reduction is applied. So we end up with a network of two inputs x_{t-1}, x_t (plus constant) and two hidden layer nodes.
Based on this $nn(3,2,1)$ network, the Lyapunov exponents of Ψ are calculated. This can be done on two ways: either along the actual data (x_t, x_{t-1}), or along a series $(\hat{x}_{t-1}, \hat{x}_t)$, where $\hat{x}_t$ is a series generated by the neural network function $nn(3,2,1)$:

$$\hat{x}_{t+1} = nn(3,2,1)(\hat{x}_{t-1}, \hat{x}_t) \tag{33}$$

The series $\hat{x}_t$ is the dynamic forecast given some initial values $\hat{x}_0 = x_0$, $\hat{x}_1 = x_1$[1]; here and in the following, such a series will be denoted as *orbit* in contrast to the actual data series $\{x_t\}$.
If the function Ψ is indeed a proper approximation of the original model then one should expect that the dynamic properties along the actual data and along the orbit data should be similar.

[1] The start values are taken from the actual data.

Table VI. Lyapunov exponents $\gamma = 4$

series	actual x_t	orbit $\hat{x}_t$
Lyapunov exponent	0.9951	1.0155
Lyapunov exponent	-0.0735	-0.0729

The results for the Lyapunov exponents are given in table (VI); they are in both cases near the theoretical values. Note that for the orbit data the largest Lyapunov exponent is greater than 1; this would correspond in the original model (27) with a coefficient of $\epsilon_t(1 - \epsilon_t)$ larger than 4: for almost all start values, the series ϵ_t would diverge. However the orbit data $\hat{x}_t$ converge to a large amplitude periodic pattern: in figure (5) the graph of $(t, \hat{x}_t)$ for the $t = 1, \cdots, 150$ (left figure), and for $t = 1, \cdots, 200$ (right figure) is compared to the graph of the actual data (t, x_t). Since the original system is chaotic, one should not expect that both series, actual and orbit data, fall together but one should that the graph of both series show a similar pattern; even in this case (and also in the larger networks get before) the orbit trajectory finally deviates essentially from the original pattern; see figure (5).

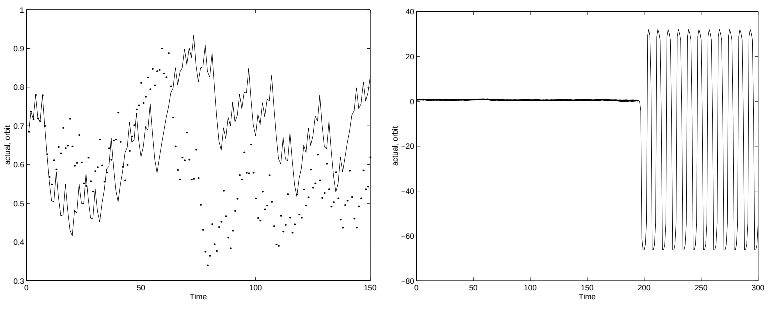

Figure 5. Actual data $CH95$, $\gamma = 4$ and orbit data (continuous line) generated by neural network $nn(3, 2, 1)$

The reason for the deviations of orbit data from the original pattern, must be found in the structural instability of the system (27) with $\gamma = 4$.

If $\gamma = 3.95$ then still a chaotic series will be generated however the system it self is structural stable. Approximating those data by a neural network, a network similar in size as before is found. In figure (6) again actual and orbit (even extended in time beyond the range of the original data) is shown. Now the *orbit* data, the dynamical forecast of

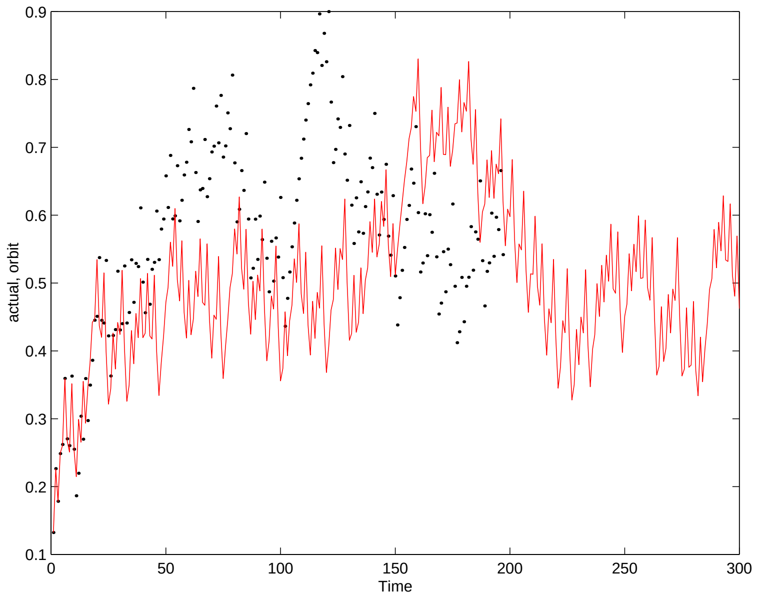

Figure 6. Actual data with $\gamma = 3.95$ and orbit data (continuous line) generated by neural network $nn(3, 2, 1)$

given value x_0, "behaves" like the original data. The same holds for the Lyapunov exponents; see table (VII)[2].

Table VII. Lyapunov exponents $\gamma = 3.95$

series	actual x_t	orbit $\hat{x}_t$
Lyapunov exponent	0.8618	0.8614
Lyapunov exponent	-0.0782	-0.0727

Finally, we look how the network $nn(3, 2, 1)$ has approximated the original data. For both cases, $\gamma = 4$, $\gamma = 3.95$, none of the parameters A, b, C and d, are very large. For instance in the case of $\gamma = 4$, the input vector of hidden nodes $xA + b$, is given as:

$$\begin{aligned}(xA+b)_1 &= -3.1557x_{t-1} + 3.3197x_t - 1.3688 \\ &= \frac{1}{3.3197}(-0.9506x_{t-1} + x_t - 0.4123) \\ (xA+b)_2 &= -0.7453x_{t-1} + 0.8095x_t + 0.0516 \\ &= \frac{1}{0.8095}(-0.9207x_{t-1} + x_t + 0.0637)\end{aligned} \tag{34}$$

[2] For $\gamma = 3.95$ no analytic value of the largest Lyapunov exponent is available; the second Lyapunov exponent will be again ~ -0.074.

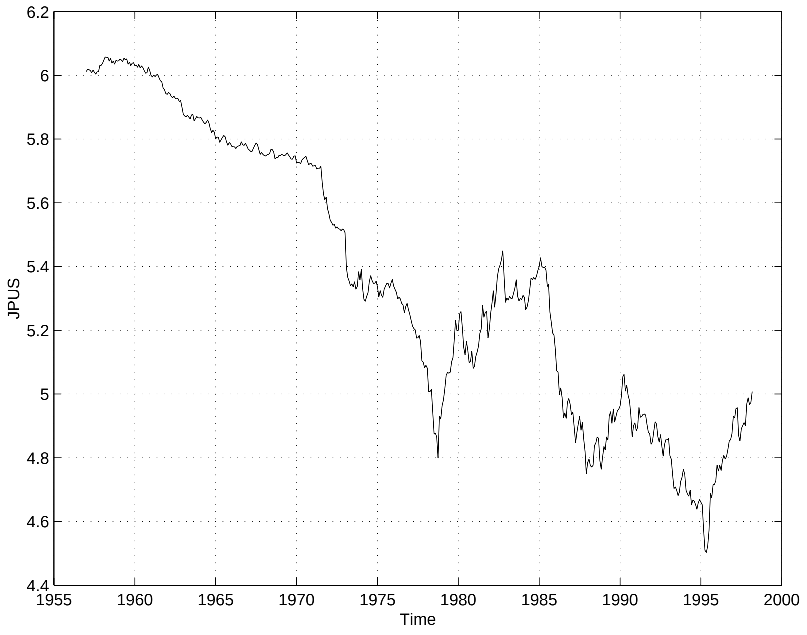

Figure 7. Time series $JPUS$: Yen-Dollar exchange rates.

Especially, the first component is remarkable: written in the variable ϵ_t, this component equals to

$$\frac{1}{3.3197}(\epsilon_{t-1} - 0.4123) \sim \frac{1}{3.3197}(4\epsilon_{t-2}(1-\epsilon_{t-2})).$$

The output of the two hidden nodes is almost equal in magnitude but do have reverse signs. Such a "behaviour" of hidden nodes, is rather common in this type of networks, and it is suggested to remove such nodes. However, that should be not applied as a "general" rule as shown here; a network with just one hidden node performs badly. In general, one should take care to remove such hidden nodes.

3.2. Nonlinear trends in real exchange rates

The economic time series are monthly observations of the natural logarithm of real exchange rates. We report those between yen and dollar (period January 1957 to March 1998). This series is denoted by $JPUS$, see figure 7. Since the original data process is unknown, a proper embedding dimension m, e.g. delay vector $(x_t, \cdots, x_{t-m+1})$, is also unknown. One way out is to extract from the correlation dimension, see Grassberger and Procaccia (1983), the embedding dimension. ; see also Kaashoek and van Dijk (1991)

Another approach would be to start with a rather large network and prune this network till further reduction will corrupt the performance essentially. In this case, the initial network was taken to be $nn(6,10,1)$ with input variables $(1, x_t, x_{t-1}, \cdots, x_{t-4})$. The performance of this network, and the successively reduced networks, are summarized in table (VIII). At the same time, in the column "λ" the largest Lyapunov exponent along the actual data set is reported while in column "$\hat{\lambda}$" the largest Lyapunov exponent along the *orbit* is reported. The applied reduction can be found in the column "Pruning".

Table VIII. Results of neural network approximation of $JPUS$ data

Network	R^2	$MSSR$	SIC	λ	$\hat{\lambda}$	Pruning
$nn(6,10,1)$	0.997	$0.16\,10^{-3}$	-3.880	0.058	-0.013	3 redundant hidden cells
$nn(6,7,1)$	0.997	$0.16\,10^{-3}$	-4.041	0.033	-0.038	2 redundant hidden cells
$nn(6,5,1)$	0.997	$0.16\,10^{-3}$	-4.130	0.048	-0.143	2 redundant input cells
$nn(4,5,1)$	0.997	$0.16\,10^{-3}$	-4.194	0.048	-0.032	2 redundant hidden cells
$nn(4,3,1)$	0.996	$0.18\,10^{-3}$	-4.188	0.076	-0.034	2 redundant input cells
$nn(2,3,1)$	0.996	$0.19\,10^{-3}$	-4.215	-0.011	-0.019	

The input variables of the resulting network $nn(2,3,1)$ are $\{(1, x_t)\}$. In figure (8) the *orbit* based on the $nn(2,3,1)$ network is shown (the data are scaled down between 0.1 and 0.9): the orbit follows nicely the pattern of the actual data converging to a scaled value of 0.26 compatible with the unscaled value of -0.74.

In all cases the orbit is stable (negative largest Lyapunov exponent) while along the actual data, the Lyapunov exponent is positive indicating an unstable series except for final the $nn(2,3,1)$ network.
The differences between the networks with respect to statistics are marginal. Note also that the Lyapunov exponent based on actual data still encompass stochastic elements if present in the data itself. The smallest network is preferable.
For a more detailed analysis on nonlinear trends in real exchange rates of several industrialized countries using neural nets we refer to Kaashoek and van Dijk (1999).

4. Phillips curve

The data are monthly US unemployment rates (all workers, 16 years and older), denoted by $LHUR$, and monthly 12-period inflation rates

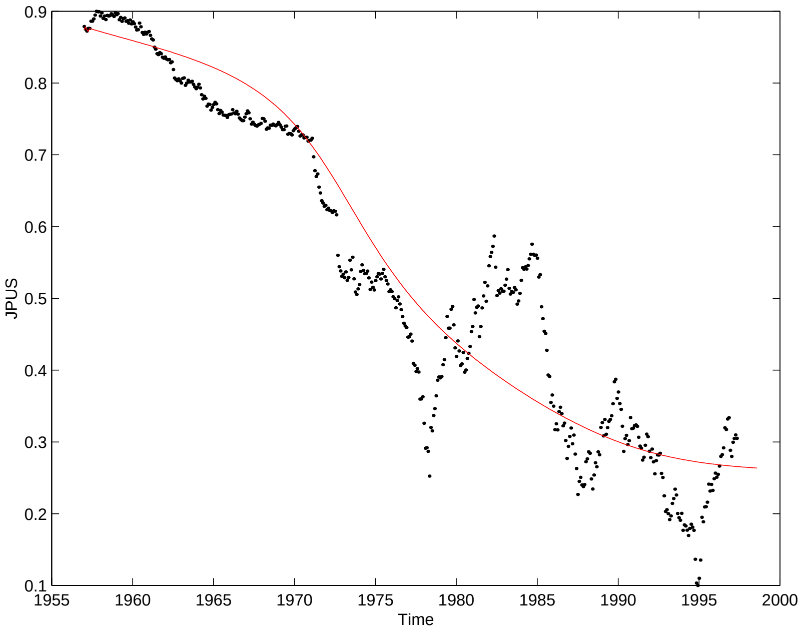

Figure 8. $nn(2,3,1)$ orbit compared to actual JPUS data

defined by $100 \ln(p_t/p_{t-12})$ in the level of the consumer price index p_t; those data are denoted by $INFR12$. The unemployment rates and price indices data start at 1960 and end at November 1997.

The Phillips curve relates unemployment rates with inflation rates; a common approach, see e.g. Sargent (1999), is to link unemployment of one year before with current inflation. In figure (9) the time series of $LHUR(-1)$[3] and of $INFR12$ are shown.

The Phillips curve data $\{(LHUR(-1)(t), INFR12(t))\}$ are shown in figure (10).

Two attempts are made to model the relation between $LHUR(-1)$ and $INFR12$.
First: Let $nn(2,6,1)$ be the neural network with inputs a constant term and the variable $LHUR(-1)$, consider 6 hidden layer cells and let the target output value be $INFR12$. The performance of this network, which is to fit the Phillips curve to the data of figure (10), is poor: $MUSSR = 0.13$, $R^2 = 0.11$. This is rather obvious because with input variable $LHUR(-1)$, the points of figure (10) can hardly to be

[3] Here, and in the following, 1 year delayed data are denoted by a -1 argument.

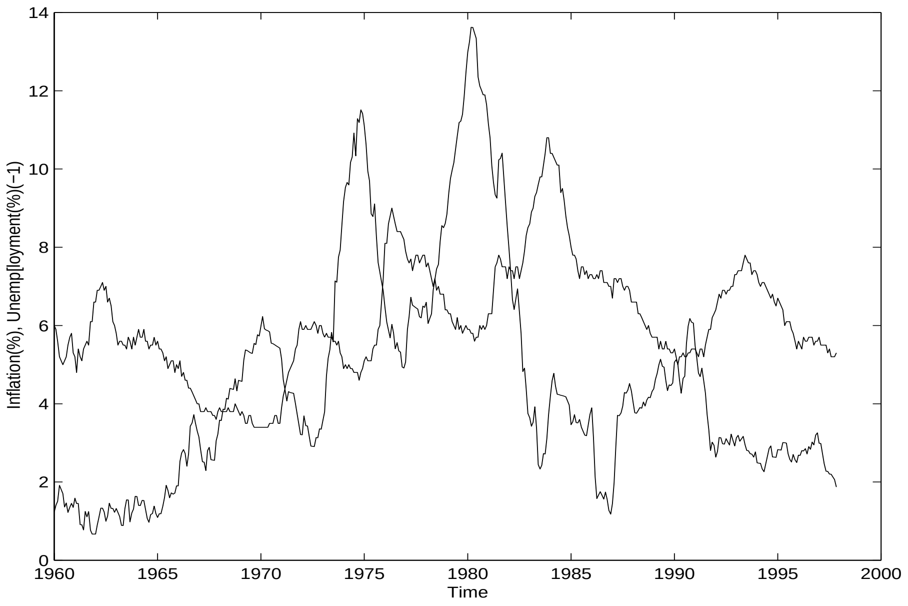

Figure 9. Time series of inflation rates $INFR12$ and unemployment rates $LHUR(-1)$

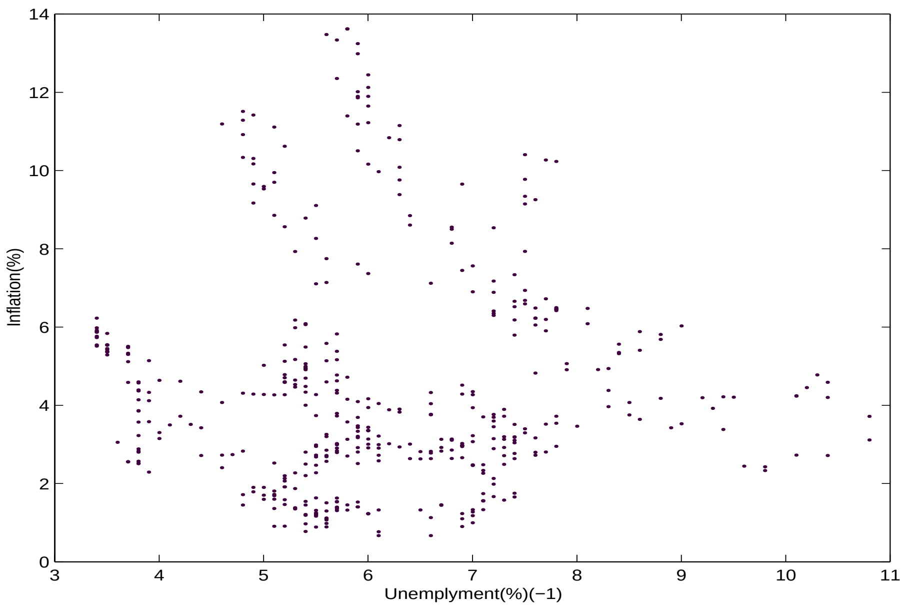

Figure 10. Phillips curve data $\{(LHUR(-1), INFR12)\}$

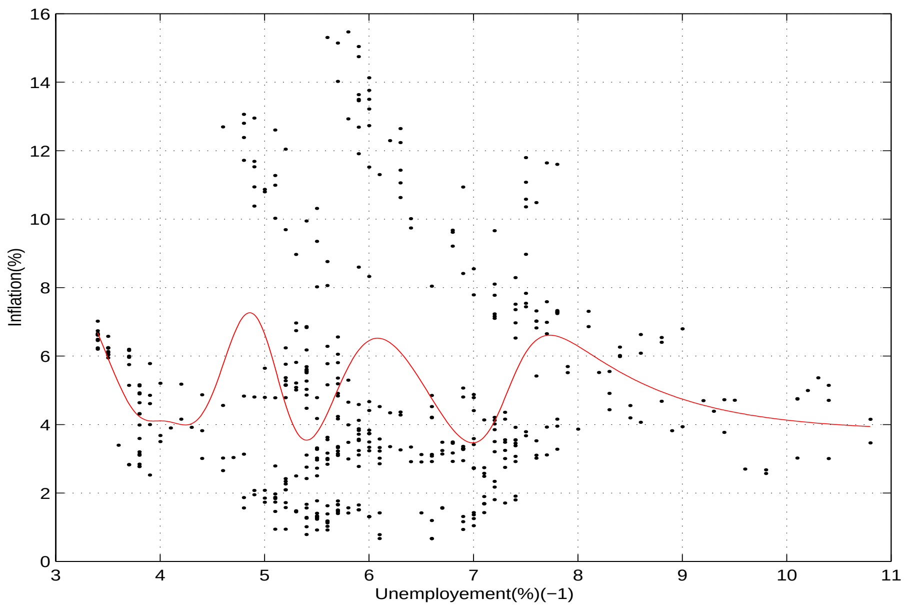

Figure 11. Graph of neural network function $nn(2,6,1)(x)$ and actual data

considered as generated by a single valued relation (function). The graph of the neural network function $x \rightarrow nn(2,6)(x)$ confirms this: it seems to consists of four descending functions (all four being standard Phillips curves) with smooth transitions in between; see figure (11). Networks with more hidden layers show similar patterns. Although a neural network is capable of generating step-functions, it can not of course model "multi-level relations". The graph in figure (10) is rather to be considered as generated by some explicit time dependent relation $(t, LHUR(-1)) \rightarrow INFR12$.

Second attempt:
Referring again to figure (11, a time varying approach is tempting as in time varying smooth transition models, see e.g. Van Dijk (1999).
Suppose 4 time periods, and for each time period a neural network approximation written as $nn_i(x), i = 1, \ldots, 4$ with

$$nn_i(x) = c_i g(a_i x + b_i) + d_i,\ \forall i. \tag{35}$$

then a formulation of time varying smooth transition model could be:

$$nn_1(x)g_1(t) + nn_2(x)g_2(t) + nn_3(x)g_3(t) + nn_4 g_4(t) \tag{36}$$

subject to some normalization, say:

$$g_1(t) + g_2(t) + g_3(t) + g_4(t) = \gamma,\ \forall t. \tag{37}$$

Hence equation (36) can be written as:

$$\begin{aligned} nn_1(x)(\gamma - g_2(t) - g_3(t) - g_4(t)) + \\ nn_2(x)g_2(t) + nn_3(x)g_3(t) + nn_4 g_4(t). \end{aligned} \tag{38}$$

Collecting the transition functions, one gets:

$$\begin{aligned} &\gamma\, nn_1(x) + g_2(t)(nn_2(x) - nn_1(x)) + \\ &g_3(t)(nn_3(x) - nn_1(x)) + g_4(t)(nn_4(x) - nn_1(x)). \end{aligned} \tag{39}$$

Now assume:

$$nn_i(x) = (1+\gamma)nn_1(x),\ i = 2, \cdots, 4 \tag{40}$$

then the model equation can be written as:

$$\gamma\{1 + g_2(t) + g_3(t) + g_4(t)\}nn_1(x) \tag{41}$$

Note that the condition (40) assumes that for all four regimes, the neural network approximation differs only by a multiplicative constant. Assuming

$$g_i(t) = c_i/(1 + \exp(-a_i t - b_i)), \tag{42}$$

then the final formulation (41) is:

$$nn(t) \times nn(x) \tag{43}$$

where $nn(t)$ is a neural network with 3 hidden layers.

Assuming 4 time transitions, our neural network model is given by:

$$(t, x) \to nn(t) \times nn(x) \tag{44}$$

with

$$nn(t) : t \to 1 + \sum_{i=1}^{3} \frac{c_i}{1 + e^{-a_i t - b_i}} \tag{45}$$

$$nn(x) : x \to \delta + \sum_{i=1}^{H} \frac{\gamma_i}{1 + e^{-\alpha_i t - \beta_i}}. \tag{46}$$

The function $nn(t)$ will be denoted as the t-network while the function $nn(x)$ will be called the x-network.

The number of hidden layer cells H in the x-network $nn(x)$ has to be defined. The input variable x will be (again) the data series $LHUR(_{-}1)$

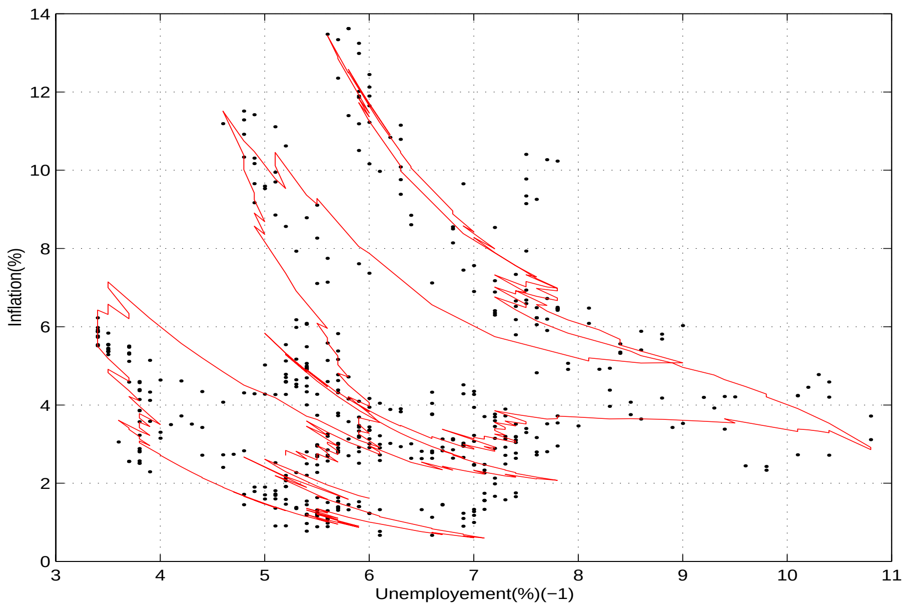

Figure 12. Estimated data of model $nn(t) \times nn(x)$, with $H = 3$, and actual data (dots)

while t will be a time-index series $\{1, 2, \cdots\}$. We report on the results for $H = 3$ and $H = 1$. Networks with more hidden layer cells didn't give other results.

Results on $H = 3$.
Statistical values: $MUSSR = 0.002$, $R^2 = 0.96$.
In figure (12) the estimated- and actual data are shown.

The performance is much better (as expected) then the foregoing network approximation.
In order to find out how the time-network performs and whether an elementary "Phillips-curve" is found by this model, the outputs of the t-network $nn(t)$ and x-network $nn(x)$ are graphed separately; see figure (13).
It seems that the x-network has indeed found some basic Phillips-curve structure. However, the t-network shows only three levels. Looking at the output of each hidden layer cell of the t-network $nn(t)$ separately reveals the same: hidden cells 1 and 2 will generate 3 levels while hidden cell 3 is almost linear in time; see figure (14)(the almost flat zero line is the figures is the graph of actual data).

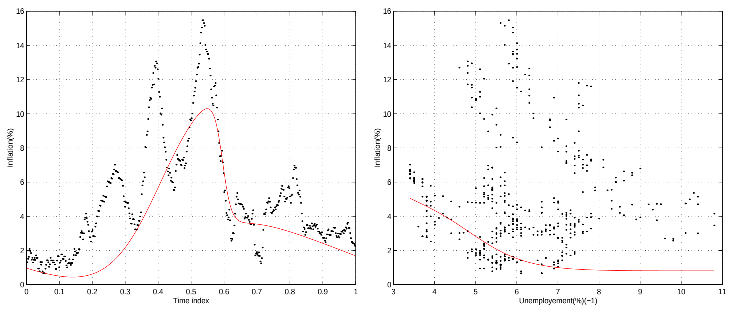

Figure 13. Output of $nn(t)$ (left) and $nn(x)$ (right) compared to actual data (dots)

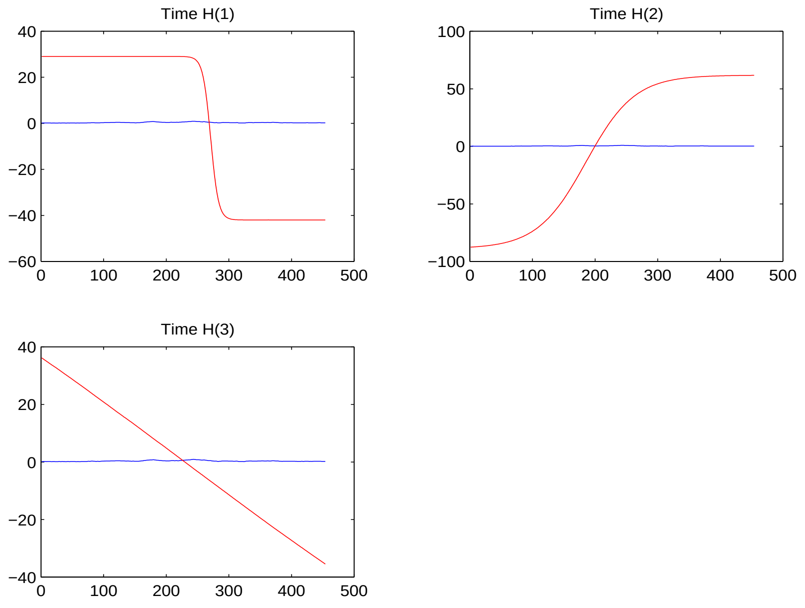

Figure 14. Output of hidden layer cells of $nn(t)$

So (at least) one transition is missing. But the x-network $nn(x)$ indicates that there is also a level transition; such a level transition is made up by two hidden cells with almost equal but reverse in sign output level. So a x-network with only one hidden cell is tried out.

Results on $H = 1$.
Statistical values: $MUSSR = 0.0023$, $R^2 = 0.93$.
In figure (15) the estimated- and actual data are shown.

The output of the t-network $nn(t)$ and x-network $nn(x)$ show the three transitions and in case of the x-network, a smooth "Phillips-curve"; see figure (16).

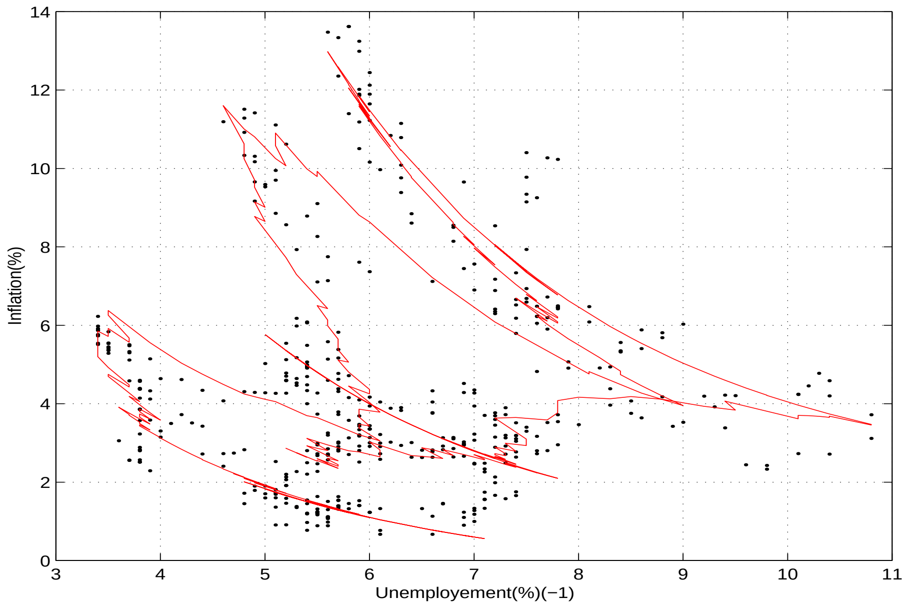

Figure 15. Estimated data of model $nn(t) \times nn(x)$, with $H = 1$, and actual data

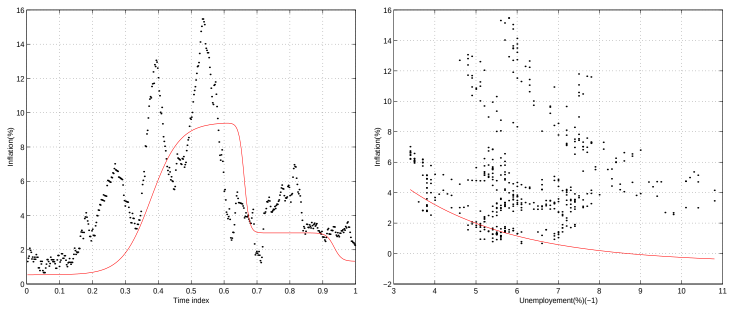

Figure 16. Output of $nn(t)$ (left) and $nn(x)$ (right) compared to actual data

It seems that the x-network has indeed found (again) some basic Phillips-curve structure and now the time-network $nn(t)$ generates indeed four levels. Approximately, those time transitions occur at $t_i = -b_i/a_i$. In this case the transitions take place at:

$$\begin{aligned} t_1 &= 171.93 \sim \text{April 1974} \\ t_2 &= 300.81 \sim \text{February 1982} \\ t_3 &= 425.13 \sim \text{June 1995} \end{aligned} \tag{47}$$

The date t_1 corresponds more or less to the first oil crises, date t_2 corresponds to a business cycle slowdown in the US economy while the third date t_3 coincides with the beginning of a period of low inflation and low unemployment.

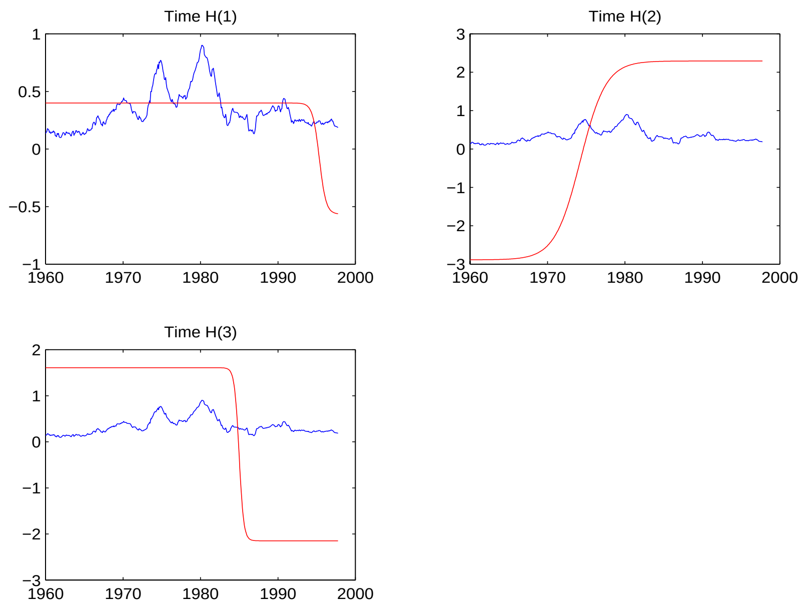

Figure 17. Output of hidden layer cells of $nn(t)$

Looking at the output of each hidden layer cell of the t-network $nn(t)$ separately reveals the same; see figure (17).

5. Conclusion

In this paper we gave a brief exposition of neural networks and their flexibility in handling complex patterns in economic data. A descriptive method to prune the size of neural nets in order to avoid overfitting is summarized. It it shown how a neural network is used to recover the dynamic properties of a nonlinear system, in particular, its stability by making use of the Lyapunov exponent. A two-stage network has been introduced where the usual nonlinear model is combined with time transitions, which may be handled by neural networks. The empirical examples on nonlinear trends in real exchange rates and a time-varying Philips curve using US data indicate the applicability of the proposed procedures. Further research is needed to allow for more than one output and it is a challenge for neural network analysis to recover common nonlinear trends in multivariate nonlinear systems.

Acknowledgements

The authors are indebted to Lennart Hoogerheide, Rutger van Oest and Daan Ooms for helpful assistance.

References

Arnold, V.I. and Avez A., *Ergodic Problems of Classical Mechanics*, Addison-Wesley Publishing Company, INC., Menlo Park, CA., 1988.

Bishop, C.M., *Neural Networks for Pattern Recognition*, Clarendon Press Oxford, 1995.

Cybenko, G., Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems*, vol.2, 1989.

Eckmann. J.P. & Ruelle, D., 1985, Ergodic theory of chaos and strange attractors, *Reviews of Modern Physics*, **57**, 617-656.

Funahashi, K., On the approximate realization of continuous mappings by neural networks, *Neural Networks*, vol.2 , 1989.

Gallant, A.R. & H. White, There exists a neural network that does not make avoidable mistakes, in *Proc. of the International Conference on Neural Networks, San Diego, 1988*, IEEE Press, New York, 1989.

Gallant, A.R. & H. White, On learning the derivatives of an unknown mapping with multilayer feedforward networks, *Neural Networks*, vol.5, 1992.

Granger, C.W.J. & T. Teräsvirta, *Modelling Nonlinear Economic Relationships*, Oxford University Press Inc., New York, 1993.

Grassberger, P. & I. Procaccia, Measuring the strangeness of strange attractors, Physica 9D. 189-208, 1983.

Guckenheimer, J. and Holmes, P., *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, Springer-Verlag, New York Berlin Heidelberg Tokyo, 1983.

Hecht-Nielsen, R., Kolmogorov mapping neural network existence theorem, in IEEE First International Conference on Neural Networks, pp.11-13, 1987.

Hecht-Nielsen, R., Theory of the Backpropagation Neural Network, in International Joint Conference on Neural Networks, pp.593-605, 1989.

Hecht-Nielsen, R., *Neurocomputing*, Addison-Wesley Publ. Co., Menlo Park, CA, 1990.

Hertz, J., A. Krogh & R.G. Palmer, *Introduction to the theory of neural computation*, Addison-Wesley Publishing Company, Reading Massachusetts, 1991.

Hornik, K., Stinchcombe, M., and White, H., Multilayer feedforward networks are universal approximators, *Neural Networks*, vol.2, 1989.

Hornik, K., Stinchcombe, M., and White, H., Universal Approximation of an Unknown Mapping and its Derivatives Using Multilayer Feedforward Networks, *Neural Networks*, vol.3, 1990.

Kaashoek, J.F. & H.K. van Dijk, A note on the detection of chaos in medium sized time series, *International Series of Numerical Mathematics*, Vol. 97, 1991, Birkhäuser Verlag Basel.

Kaashoek, J.F. & H.K. van Dijk, Evaluation and Application of numerical procedures to calculate Lyapunov exponents, *Econometric Reviews*, special issue, Vol. 13, No.1, 1994.

Kaashoek, J.F. & H.K. van Dijk, A simple strategy to prune neural networks with an application to economic time series, Report 9854/A, Econometric Institute Erasmus University Rotterdam, 1998.

Kaashoek, J.F. & H.K. van Dijk, Neural network analysis of varying trends in real exchange rates, Report EI9915/A,, Econometric Institute Erasmus University Rotterdam, 1999.

Kolomogorov, A.N., On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition, American Mathematical Montly Translation, vol.28, 1963. (Russian original in Doklady Akademii Nauk SSSR, 144.)

Leshno, M., Lin, V. Y., Pinkus, A. & Schocken, S., Multilayer Feedforward Networks With a Nonpolynomial Activation Function Can Approximate Any Function, *Neural networks*, vol.6, 1993.

Malinvaud, E., *Statistical Methods of Econometrics*, North-Holland Publ. Co., Amsterdam London, 1970.

Mozer, M.C. & P. Smolensky, Skeletonization: a technique for trimming the fat from a network via relevance assessment, in D.S. Touretzky (Ed.) *Advances in Neural Information Processing Systems*, vol. 1, San Mateo, CA., 1989.

Press, W.H., B.P. Flannery, S.A. Teukolsky & W.T. Vetterling, *Numerical Recipes*, Cambridge University Press, Cambridge, 1988.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J., Learning representations by error propagation, in D.E. Rumelhart, J.L. McClelland and the PDP research Group (Eds.) *Parellel distributed processing, vol. 1.*, Cambridge, MA, MIT Press, 1986.

Sargent, T.J., *The Conquest of American Inflation*, Princeton University Press, Princeton, NJ, 1999.

Simpson, P.K., *Arificial neural systems: foundations, paradigms, applications and implementations*, Pergamon Press, 1990.

Takens, F., Detecting strange attractors in turbulence, in D.A. Rand and L.S. Young (eds.), *Dynamical systems and turbulence*, Springer-Verlag, Berlin, 1981.

Theil, H., *Principle of Econometrics*, Wiley & Sons, 1971

Van Dijk, D., *Smooth Transition Models: Extensions and Outlier Robust Inference*, Tinbergen Institute Research Series, no.200, Erasmus University Rotterdam, 1999.

White, H., Some Asymptotic Results for Learning on Single Hidden Layer Feedforward Network Models, *Journal of the American Statistical Association*,vol.**84**, no.408, 1989.