The Use of Feed-forward and Recurrent Neural Networks for System Identification

T. L. Burrows & M. Niranjan

CUED/F-INFENG/TR158

December 1993

Abstract

Neural networks with nonlinear sigmoid functions at the hidden nodes have been shown to give improved performance over linear models for time series prediction problems. We demonstrate that whenever the network produces a useful solution to this problem, the hidden nodes operate predominantly in the linear region of their sigmoid function, and that small excursions into the nonlinear region allow improved prediction. Using several nonlinear time series, we demonstrate that this allows us to exploit standard linear system identification techniques. For a speech prediction problem, we compare the performance of a feed-forward and recurrent architecture and in view of our observations, attribute the improved performance of the recurrent network to a parameter estimation based on *output error* minimisation, rather than the *equation error* minimisation performed by feed-forward networks and linear prediction analysis.

Cambridge University Engineering Department
Trumpington Street
Cambridge CB2 1PZ
England

Email: tb119/niranjan@eng.cam.ac.uk

Contents

1 Introduction

The underlying mechanisms that generate many time series are nonlinear. Speech data is a typical example, in which nonlinearities are introduced by rapid segment transitions during and between phonemes, turbulent excitation during unvoiced segments and glottal opening/closure during pitch periods of voiced speech. Good prediction of such data relies on the identification of the system from which the data is generated. Linear models of the system have a limited performance since they may not capture the structure of the data or the underlying system dynamics. Nonlinear models are needed and studies by many authors (Lapedes and Farber, 1987; Tishby, 1990; Wu and Fallside, 1992) have shown that these may be implemented as neural networks.

For neural network-based models to be applicable to continuous, real-valued signals, they must be able to represent continuous functions, despite saturating functions in the hidden nodes. Universal approximation properties of neural networks (Cybenko, 1989; White, 1989) have shown that even feed-forward networks with a single layer of hidden nodes with arbitrary continuous sigmoid functions can approximate a continuous function with arbitrary precision, provided no constraints are placed on the number of hidden nodes or the size of their weights, and that the network complexity increases at a rate approximately proportional to the size of the training data.

Simulations have shown that neural networks with a single hidden layer of nonlinear nodes have the power to model complex nonlinear mechanisms (Lapedes and Farber, 1987; Narendra and Parthasarathy, 1990) and can improve on linear models by allowing the generation of limit cycles (Chen et al., 1990). The application of nonlinear neural network-based models to the prediction of speech has shown an improvement in prediction gain over linear models of 2–3dB (Tishby, 1990; Townshend, 1991). Recurrent neural networks, in which feedback around hidden nodes is introduced, improve modelling of the dynamics of a signal (Back and Tsoi, 1991b) and have also been shown to give improved prediction gains over linear prediction techniques (Wu and Fallside, 1992).

Much insight into the operation of neural networks has been gained from examination of the responses of the hidden nodes. Previous work has concentrated on the operation of the hidden nodes of networks for classification tasks (Webb and Lowe, 1990) and multilayer perceptrons (Bourland and Kamp, 1988; Gallinari et al., 1988). Research on system identification using single hidden layer recurrent networks (Back and Tsoi, 1991a), has shown that each node attempts to model the pole/zero dynamics of the system. In this paper we show that for a single hidden layer neural network to produce a good solution to system identification tasks associated with *continuous, real-valued* time series, the hidden nodes operate predominantly in the linear region, with small excursions into nonlinear regions, and little hard saturation. We illustrate this with simulations for the prediction of artificial time series data (Chen et al., 1990) and sunspot data (Priestley, 1988), using a single hidden layer feed-forward network. To demonstrate the usefulness of our observation, we investigate the operation of feed-forward and recurrent neural networks for the prediction of speech phonemes from the TIMIT database (Garofolo, 1988), and explain their different performance properties in terms of their linearity, by drawing parallels with linear system identification theory.

The paper is organized as follows. In section 2, we describe the structure of a single hidden layer network model and explain the behaviour of the hidden nodes during system identification. In section 3 we give illustrations of the responses of hidden nodes for several different simulations and network structures with a single hidden layer. In sections 3.1 and 3.2, following the simulations of Chen (Chen et al., 1990), we use a feed-forward network for the prediction of artificial time series data and sunspot data. In section 3.3, we use both feed-forward and recurrent networks for the prediction of speech phonemes. In section 4 we demonstrate how our observation of linearity of the hidden nodes allows us to exploit linear system identification theory. We use two specific examples : the initialisation of network weights to those of an equivalent linear model and a comparison of the parameter estimation techniques of feed-forward and recurrent networks. In section 5 we summarise our conclusions.

2 Neural Networks for System Identification

System identification is the term applied to the development of mathematical models (parametric forms of functional mappings) between the outputs and inputs of a dynamical system, based on observations of this data. We can regard the training of a neural network as essentially the same problem : identifying the underlying functional mapping between the inputs and outputs of a system. For parametric models, system identification is traditionally based on estimation of parameters of *linear* models by minimising some norm of the prediction error. Typically a quadratic norm is used, giving rise to a least squares scheme analogous to back-propagation. Once the model parameters have been evaluated, the model may be used to predict the system output for some unseen input, or it may simply be driven by its own predicted output, starting from some initial condition.

2.1 Structure of the Neural Network Model

The basic structure of a neural network-based model is shown in Fig. 1, in which square boxes represent unit time delays. A single linear output node allows predictions of any magnitude and bias terms may be present at the hidden nodes, which are interpreted as a fixed input of magnitude 1.

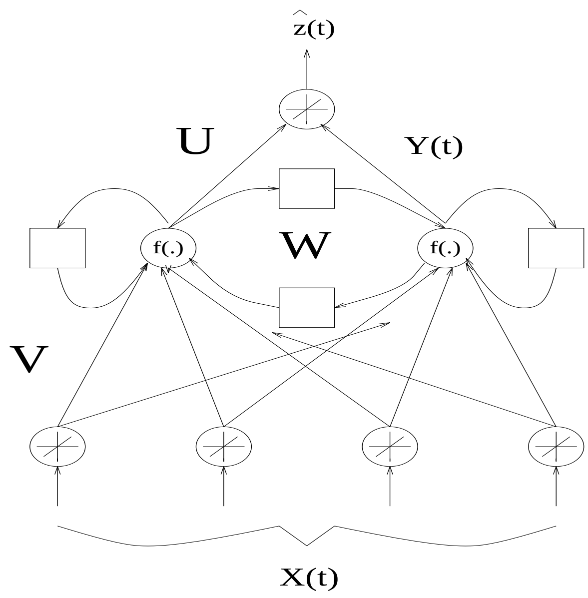


Figure 1: Structure of a neural network model

The model has a single hidden layer with n_h nodes and an input layer with n_i nodes. The input vector, $X(t)$, is fed via weight matrix V, to the hidden nodes. The output of the hidden nodes, $Y(t)$, is fed via weight matrix U, to form the predicted output, $\hat{z}(t)$, after passing through a differentiable nonlinear function $f(.)$. For a recurrent network, there is also feedback of the hidden node output via weight matrix W. The total number of network parameters is given by n_θ. The model is described by (1) and (2)

$$Y(t) = f(VX(t) + WY(t-\tau)) \qquad (1)$$

$$\hat{z}(t) = UY(t) \qquad (2)$$

A simple feed-forward network is represented by setting $W = 0$. If $n_h = n_i$, $V = I$, $f(X) = X$, $W = 0$, where I is the identity matrix, then the network reduces to a linear model, $\hat{z}(t) = UX(t)$.

For a target output $z(t)$, training by back-propagation minimises the mean squared prediction error, E, over an N–length sequence (3)

$$E = \frac{1}{N}\sum_{t=1}^{N}\big(z(t) - \hat{z}(t)\big)^2 \tag{3}$$

For *One Step Ahead* prediction (OSA), the input vector, $X_{\mathrm{OSA}}(t)$, consists of delayed samples of the target output, $z(t)$, delayed samples of an input sequence, $u(t)$, and a 1 to represent the bias terms, as in (4). For a time series model, the input terms, $u(t\ldots)$, of $X_{\mathrm{OSA}}(t)$ are set to zero. Following Chen (Chen et al., 1990), we define an input vector, $X_{\mathrm{MDO}}(t)$, for *Model Deterministic Output* (MDO), which consists of delayed samples of the predicted output vector, $\hat{z}(t)$, delayed samples of an input sequence, $u(t\ldots)$, and a 1 to represent the bias terms, as in (5). For the system unforced response, the input terms of $X_{\mathrm{MDO}}(t)$ are set to zero, hence the MDO and unforced response of a time series model are equivalent, and for a linear model give the impulse response. The number of input nodes is $n_i = n_u + n_z + 1$.

$$\begin{aligned} X_{\mathrm{OSA}}(t) &= [z(t-1), z(t-2), \ldots, z(t-n_z), u(t-1), u(t-2), \ldots, u(t-n_u), 1]^{\mathrm{T}} \qquad (4)\\ X_{\mathrm{MDO}}(t) &= [\hat{z}(t-1), \hat{z}(t-2), \ldots, \hat{z}(t-n_z), u(t-1), u(t-2), \ldots, u(t-n_u), 1]^{\mathrm{T}} \qquad (5)\end{aligned}$$

The performance of the networks is measured by the prediction gain in decibels, which for an N–length data sequence is

$$\text{Prediction Gain} = \log_{10}\left(\frac{\sum_{t=1}^{N} z^2(t)}{\sum_{t=1}^{N}\big(z(t) - \hat{z}(t)\big)^2}\right) \tag{6}$$

2.2 Behaviour of Hidden Nodes of Single Hidden Layer Network Models

For continuous, real-valued input and output data, a system model must be able to represent a continuous function. In a neural network-based model, the hidden nodes contain nonlinear functions which usually have saturation regions. A hidden node operating in these saturation regions has a constant output for a given input range. In a single hidden layer network, this output simply contributes a bias to the overall network output. If the weights of the hidden layer cause all the hidden nodes to operate in saturation, the network output will be discrete. For sigmoid functions, each saturated node has a binary output giving a discrete network output of 2^{n_h} levels for all n_h hidden nodes in saturation. The network thus operates as an anologue-to-digital converter, a situation which can easily arise at the onset of training due to poor initialisation of the network weights.

If no constraints are applied to the size of the weights of the hidden layer, such that all nodes are driven into saturation, the network can only approximate a continuous function well if the number of hidden nodes is large. Universal approximation properties for single hidden layer networks (Cybenko, 1989; White, 1989) have shown that for an arbitrary sigmoid function and no constraints on the size of the weights, approximations to continuous, real-valued functions are possible provided no constraints are placed on the number of hidden nodes. However, small networks with few hidden nodes are more desirable since they are trained more rapidly, require less training data, are easier to analyse and are potentially useful in data reduction applications such as low bit-rate coding. If we constrain our network model to have a single hidden layer with a small number of nodes, the network will form a good model of a continuous, real-valued function if the hidden nodes operate predominantly in the linear region of the sigmoid function. As the number of hidden nodes increases, saturation of some nodes may occur without losing the continuous nature of the model output, provided some nodes remain linear. Ideally, we require the model weights to converge to a solution which represents the underlying mechanism which produced the data, for example to generate limit cycles when driven by the model output (if these exist). A linear model cannot do this, since driving the model by its own output simply reproduces the impulse response of the linear system, which decays to zero. Hence, the weights of the hidden nodes must permit small excursions into the slightly nonlinear, but unsaturated regions of the sigmoid function. This may introduce sufficient nonlinearity into the model to allow generation of limit cycles. Nonlinear regions of operation may therefore improve the prediction capability of the neural network-based model over that of a linear model.

In the following section, we illustrate these observations by applying the neural network model of Fig. 1 to the identification of a predictor for both simulated and real data, and give examples of the hidden node responses.

3 Simulation Studies

We used single hidden layer networks for simulation of artificial and real data and examined the operation of the hidden nodes. In section 3.1 we used a feed-forward network to predict some artificial nonlinear data suggested by Chen (Chen et al., 1990), in section 3.2 we used a feed-forward network for the prediction of sunspot data (Priestley, 1988) and in section 3.3 we used both feed-forward and recurrent networks for the prediction of speech phonemes from the TIMIT database (Garofolo, 1988).

3.1 Prediction of Simulated Nonlinear Data

We used the simulated data system of the first simulation study suggested by Chen (Chen et al., 1990), which consists of 500 data points generated by

$$\begin{aligned} z(t) \;=\; & \Big(0.8 - 0.5\exp\big(-z^2(t-1)\big)\Big)z(t-1) - \Big(0.3 + 0.9\exp\big(-z^2(t-1)\big)\Big)z(t-2) \\ & + u(t-1) + 0.2u(t-2) + 0.1u(t-1)u(t-2) + e(t) \end{aligned} \tag{7}$$

where the system noise, $e(t)$, is a Gaussian white sequence with zero mean and variance 0.04 and the system input, $u(t)$, is an independent sequence of uniform distribution with mean zero and variance 1.0. A feed-forward net of the structure used by Chen *et al.* (hidden nodes with sigmoid nonlinearities and bias inputs, $n_u = 2, n_z = 2, n_h = 5$ ($n_\theta = 30$)), was trained by back-propagation. For comparison, we also estimated, by a least squares method, a linear ARX model (Ljung, 1987), given by

$$\hat{z}(t) = \sum_{i=1}^{n_b} b_i u(t-i) - \sum_{j=1}^{n_a} a_j z(t-j) \tag{8}$$

For $n_b = n_u$, $n_a = n_z$, the linear model is of the same form as the data system and the feed-forward network model since it has a regression of both the output data (autoregressive or AR model) and the input data (exogeneous input or X model).

We trained ten networks from different random weight initialisations and examined the hidden node responses for both OSA and MDO inputs. Typical outputs of the feed-forward network and linear ARX model are shown in Fig. 2 and the corresponding hidden node responses in Fig. 3. We found that for both OSA and MDO inputs, hidden nodes operated mainly in the linear region of the sigmoid function, for example hidden node responses 1–4 of Fig. 3. Small excursions into the curved regions introduced nonlinearity into the feed-forward network. As shown by the mean prediction gains of Table 1, this allowed the feed-forward network to outperform the linear model for both OSA and MDO inputs. In the example of Fig. 3, saturation of node 5 was tolerated without the loss of a continuous network output, due to the continuous output from the other nodes.

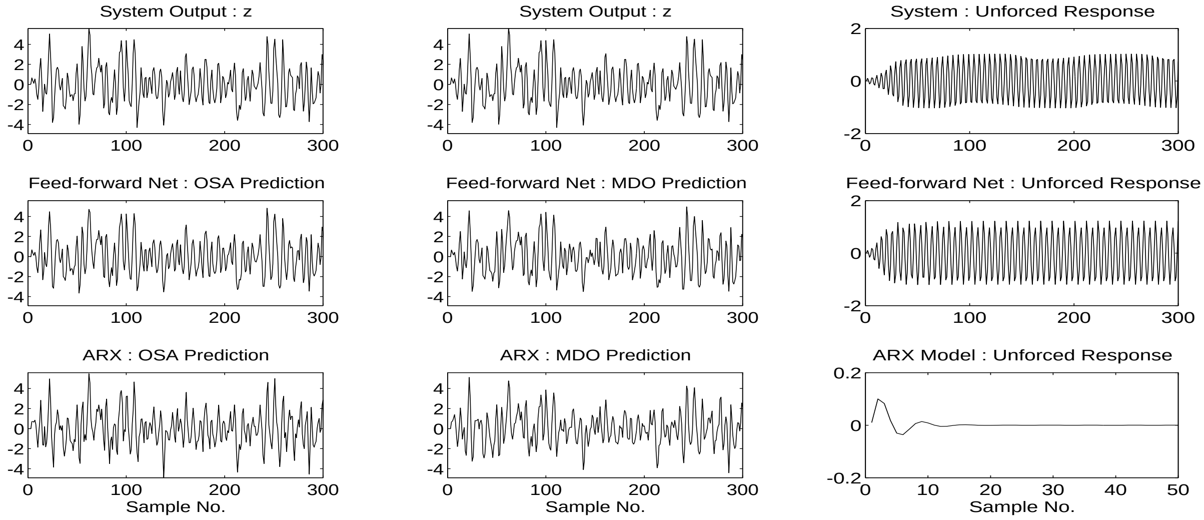


Figure 2: Model predictions for OSA and MDO inputs, unforced system and model responses (initial condition: $z(-1) = 0.01$, $z(0) = 0.1$) for simulated nonlinear data

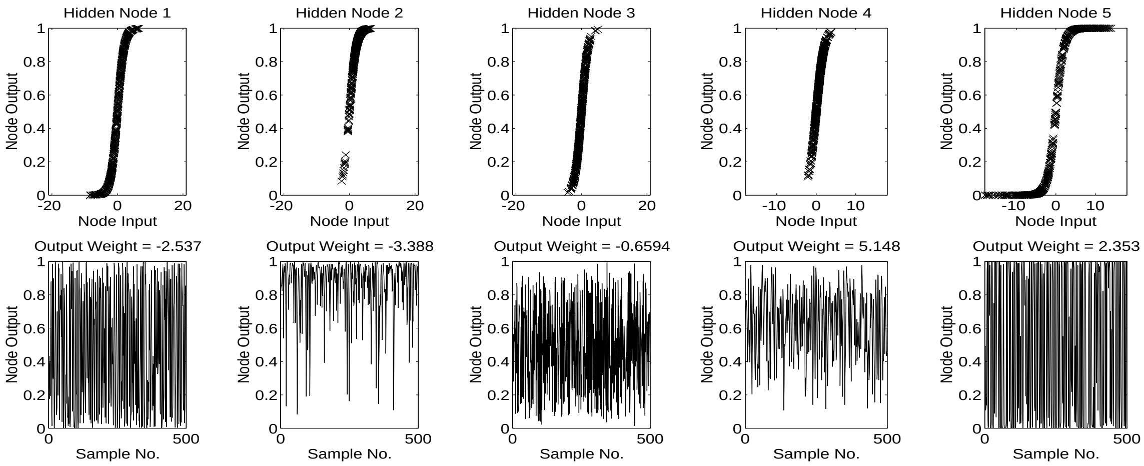
	ARX	Feed-Forward Net	AR	Feed-Forward Time Series
Mean OSA	10.89	16.91	15.35	16.38
Mean MDO	8.66	10.84		

Table 1: Prediction performance (in dB) for simulated nonlinear data

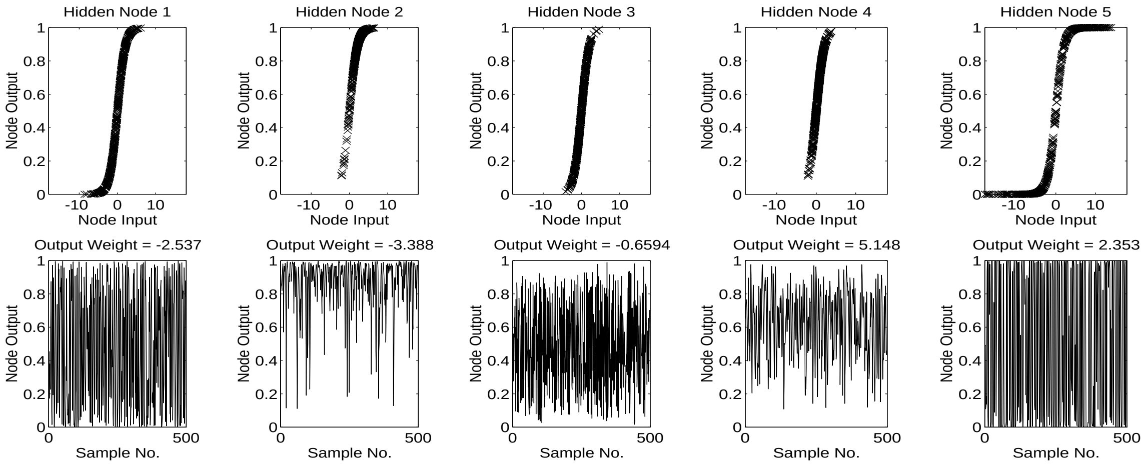
To determine the ability of the model to capture the underlying mechanism which generated the data, we examined the unforced model response (network input $X_{MDO}(t)$ with $u(t) = 0$). The unforced responses of the system ($e(t) = 0$, $u(t) = 0$), an example feed-forward network and the ARX model, starting from the same initial condition, are shown in Fig. 2. The unforced system response was a stable limit cycle. The linear model gave a decaying response but the network model was able to generate a limit cycle of approximately the same period as that of the system, although of a different shape.

We used the system unforced response of Fig. 2 to train a network with $n_z = 2$, $n_u = 0$, $n_h = 10$ ($n_\theta = 40$). This is the same form as a linear AR (autoregressive) time series model (Ljung, 1987), which can be represented by (8) with $b_i = 0$ for all i. We compared the OSA prediction and the unforced response with that of a linear AR model of order $n_z = 2$, estimated by the 'Burg' algorithm (Schafer and Rabiner, 1978). An example OSA prediction and unforced response are shown in Fig. 4. It was again found that the hidden nodes operated predominantly linearly, excursions into the nonlinear regions allowing the generation of a limit cycle with a closer resemblance to the original system limit cycle than that of the previous model.

These results show that a nonlinear model based on a feed-forward network can outperform a linear model of the same form for prediction of nonlinear data. We observe that the hidden nodes operate predominantly in the linear region of the sigmoid function. For good MDO performance however, we require the model to capture the dynamics of the underlying mechanism which generates the data. Permitting small excursions into the nonlinear regions of the sigmoid function introduces nonlinearity into the model which allows the generation of limit cycles. Hence, we gain by using a nonlinear model, although the hidden nodes operate predominantly linearly.



a) One-step ahead (OSA) prediction



b) Model deterministic output (MDO)

Figure 3: Hidden node responses for feed-forward network in one-step ahead and model deterministic output prediction of simulated nonlinear data

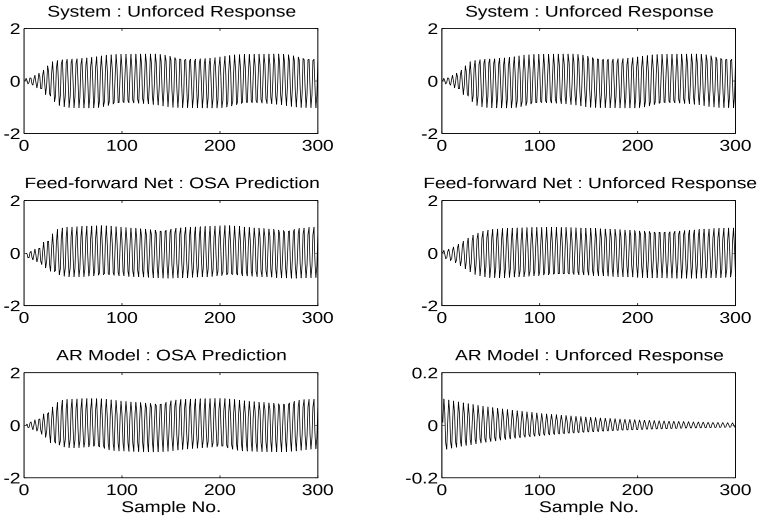


Figure 4: Unforced system and time series model responses for simulated nonlinear data (initial condition: $z(-1) = 0.01$, $z(0) = 0.1$)

3.2 Prediction of Sunspot Data

We used the annual sunspot time series data (Priestley, 1988) as our second simulation, which was also studied by Chen (Chen et al., 1990). This data has a cyclic nature with an approximate period of 11 years. We fitted a feed-forward network with sigmoid nonlinearities and $n_z = 9$, $n_u = 0$, $n_h = 5$ ($n_\theta = 55$) to the first 221 data points, which were normalised to lie in the range 0—1. Several such networks were trained from different random weight initialisations. For comparison, we also estimated a linear AR model of order $n_z = 9$, by the 'Burg' algorithm.

The OSA prediction and unforced response of an example network and the linear AR model are shown in Fig. 5, and the hidden node responses for prediction and unforced response are shown in Fig. 6. The OSA prediction gain for the illustrated network was 12.04 dB compared to 11.24 dB for the linear AR model. It was observed that for all networks generated, the hidden nodes operated predominantly in the linear region for both OSA prediction and unforced response. Excursions into the nonlinear regions of the sigmoid function, as in node 2 of Fig. 6 for example, allowed the generation of limit cycles. The example limit cycle in Fig. 5 has a similar period to that of the sunspot data, and suggests that the nonlinearity introduced into the feed-forward network model has increased the ability to capture the underlying dynamics of the system. The average OSA prediction gain (for ten networks) was lower than that for the illustrated network, at 11.72 dB. This was due to the fact that many of the networks operated almost exclusively in the linear region and did not generate a limit cycle. Examination of the prediction gain per training cycle suggested that the networks had not yet converged to a minimum solution.

3.3 Prediction of Speech Data

For this simulation, we selected five examples of the phonemes 'ix', 'n' and 's', for various speakers and dialects from the TIMIT database (Garofolo, 1988). Each example consisted of 800 samples for phonemes 'ix' and 's', and 400 samples for phoneme 'n'. The data was normalised to lie in the range $-1 \ldots 1$, half the samples of each segment being used for training and half for testing. These phonemes provide examples of voiced, unvoiced and nasal sounds which have very different spectral structures. For each phoneme example, we trained a recurrent network with a full W weight matrix, Fig. 1, and a feed-forward network with the same number of parameters, n_θ. We used the nonlinear function $f(x) = \tanh(x)$ in a single hidden layer of nodes with no bias terms. For comparison, linear networks with equal numbers of parameters were also trained on all segments. We averaged

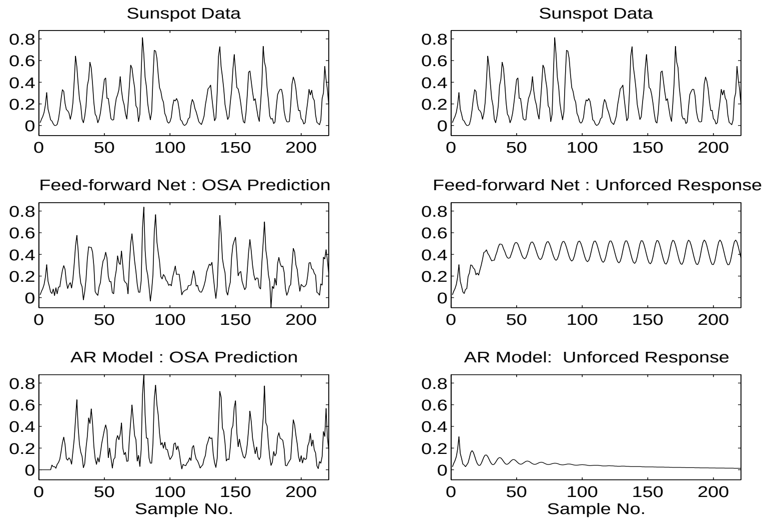


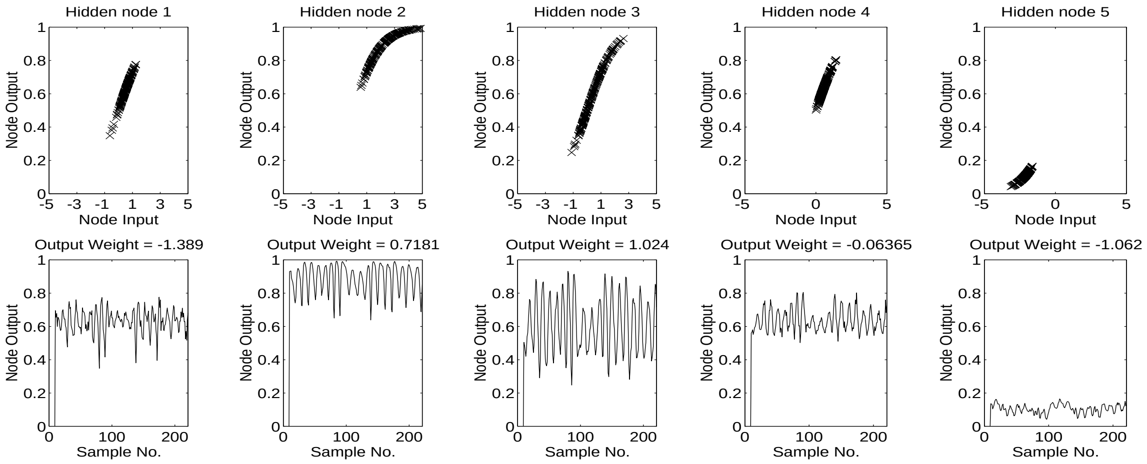
Figure 5: One-step ahead (OSA) prediction and unforced responses for sunspot data

the network performance over all test segments of each phoneme, over all phonemes for structure 1, and over all structures for one example of phoneme 'ix'. Three structures of networks were trained as shown in Table 2.

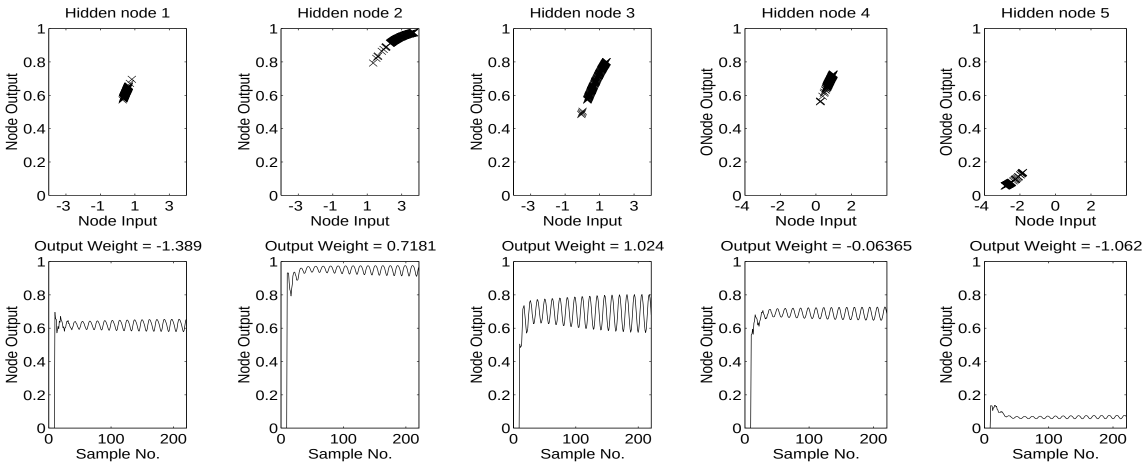
The trained networks were used to predict the test segments and the operation of the hidden nodes was examined. It was found that for all structures during testing, the hidden nodes of both the recurrent and the feed-forward network operated predominantly in the linear region of their tanh function, with small excursions into curved regions. Examples of the performance of recurrent, feed-forward and linear networks of structure 1 are shown in Fig. 7. For these networks, the prediction gains in training were 15.18, 13.71 and 8.65 dB for recurrent, feed-forward and linear networks respectively. The prediction gains in testing were 13.69, 11.85 and 13.31 dB respectively.

We also examined the unforced response of the trained networks using the first few samples of the training segments as an initial input vector. It was found that for networks operating almost exclusively in the linear region, such as those illustrated in Fig. 7, no limit cycle was developed. However, networks with excursions into the nonlinear regions allowed the generation of a limit cycle. Fig. 8 shows an example in which the recurrent network generated a limit cycle of a similar period to the speech, but the feed-forward network trained on the same data did not. For these networks, the prediction gains in training were 12.69, 13.30 and 13.49 dB for recurrent, feed-forward and linear networks respectively. In testing however, the performance of the recurrent network improved on that of the feed-forward and linear network, with prediction gains of 12.36, 11.21 and 10.11 dB respectively. In section 4.2, we show that this is due to the fact that the recurrent network is better able to capture the underlying dynamics of the system which generates the data and is hence more capable of generalisation.

The average prediction gains for training and testing are summarised in Table 3. For phoneme 'ix', comparison of structures 1 and 2 shows worse performance for structure 2, which has more input nodes. Examination of the unforced response of these models showed that in most cases they generated a limit cycle with a poor periodic match to that of the speech. This suggests that structure 2 is a poor model structure for the underlying mechanism generating the data. Structure 3 shows a similar average performance to structure 1 on phoneme 'ix', despite the presence of an additional hidden node. Comparison of the average performance on the different phonemes shows better performance of all models on the voiced phonemes 'ix' and 'n', which have periodic properties, than on the unvoiced phoneme 's', which is a much harder nonlinear problem. The average network performances show that both feed-forward and recurrent networks can outperform a linear network with the same number of parameters by up to 2dB, which is in agreement with the results of Tishby for speech data (Tishby, 1990). They do not yield conclusive evidence as

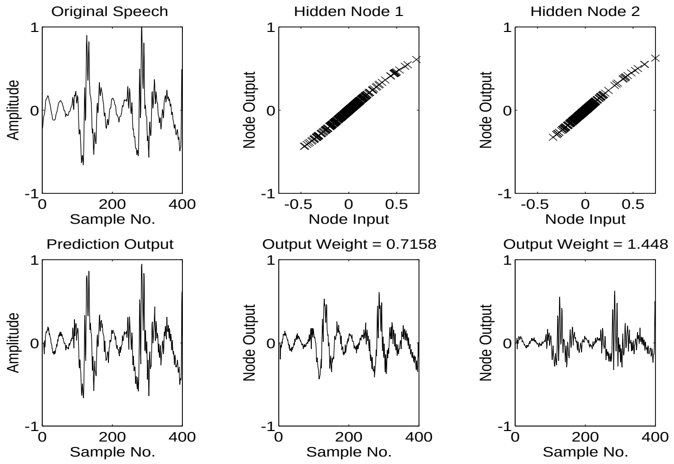


a) One-step ahead prediction

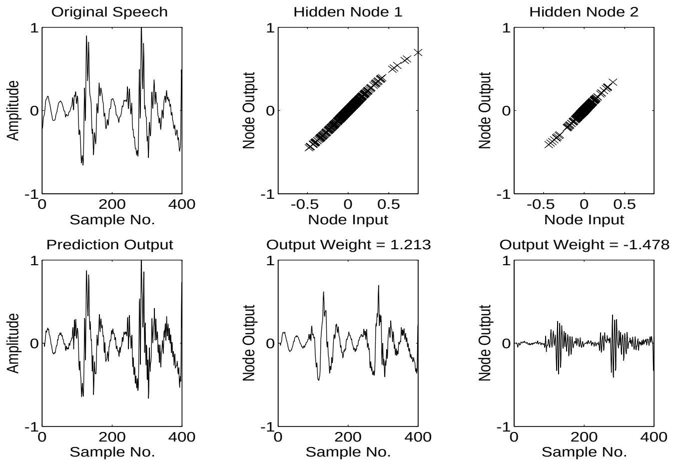


b) Unforced response

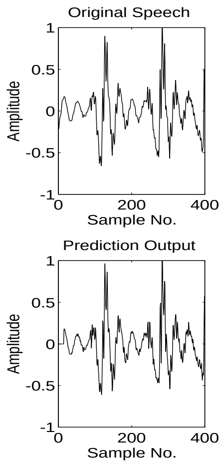
Figure 6: Hidden node responses for feed-forward network in one-step ahead prediction and unforced response for sunspot data



a) Recurrent network

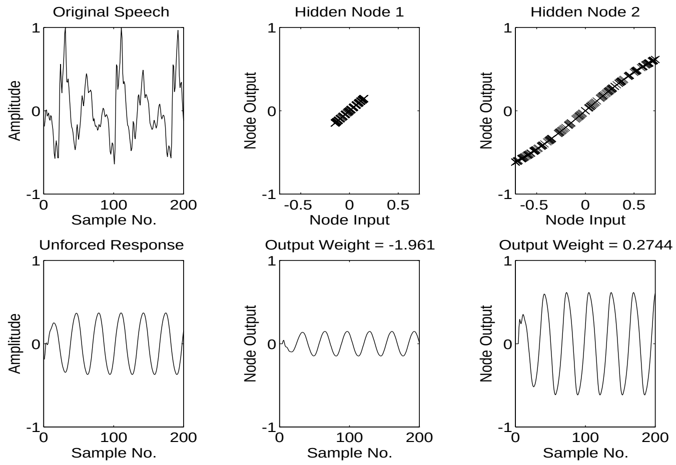


b) Feed-forward network

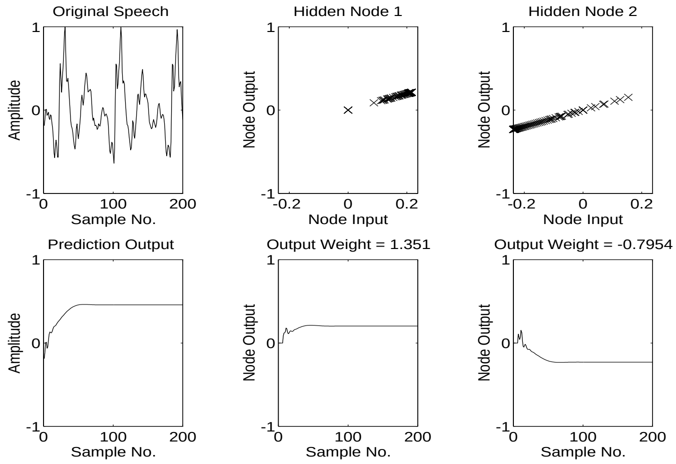


c) Linear network

Figure 7: Prediction performance and hidden node responses for test phoneme 'ix2'



a) Recurrent network



b) Feed-forward network

Figure 8: Unforced response for networks trained on phoneme ‘n’

Structure	Type	n_i $(= n_z)$	n_h	n_o	n_θ
1	R	4	2	1	14
	F	6	2		
	L	14	14		
2	R	10	2	1	26
	F	12	2		
	L	26	26		
3	R	5	3	1	27
	F	8	3		
	L	27	27		

Table 2: Network structures for speech data (R: recurrent, F: feed-forward, L: linear)

Data	Structure	SNR Training (dB)			SNR Test (dB)		
		rnn	fnn	ln	rnn	fnn	ln
s	1	5.88	5.87	6.05	5.34	5.39	5.49
n	1	12.98	13.02	12.39	11.63	11.67	9.40
ix	1	13.21	12.64	12.08	11.86	11.41	11.31
Average over 15 phonemes : structure 1		10.24	10.08	9.75	9.27	9.13	8.41
ix	2	10.81	9.89	9.76	10.11	8.98	9.04
ix	3	12.42	12.36	9.41	10.25	11.43	8.61
Average over all structures for 'ix' : 15 examples)		12.15	11.63	10.42	10.74	10.61	9.65

Table 3: Prediction performance for phonemes from the TIMIT database

to the relative performance of recurrent and feed-forward networks. This may be due to unfair comparison tecniques (for example different predictive orders), poor weight initialisations or insufficient training examples. A closer examination of the performance of the networks showed that when both recurrent and feed-forward networks operated predominantly *linearly*, a recurrent network outperformed a feed-forward network with the same number of parameters. In the following section, we draw on the theories of linear system identification to explain this increased modelling power in terms of an estimation of the parameters by output error minimisation, rather than the equation error minimisation performed by feed-forward and linear networks.

4 Exploiting Linear System Identification

An overview of the connections between system identification and neural networks is given by Ljung (Ljung and Sjöberg, 1992), in which a neural network is considered as just another type of model for approximating the functional relationship between some input and output data. This opens up a range of well-established theories with which to study the problems of model validation, observability and identifiability, suitable model structure and complexity and parameter bias for a given problem. Many of these system identification techniques have been applied successfully to neural networks (Chen et al., 1990; Billings et al., 1992) and have also led to the development of alternative training methods, such as the Recursive Prediction Error Method (Chen et al., 1990). Clearly, the observation that the nodes of a single hidden layer network operate predominantly in the linear region is useful since it allows parallels to be drawn with *linear* system identification. We illustrate the application of linear ideas to neural networks with two specific examples : initialisation of network weights and comparison of parameter estimation by feed-forward and recurrent networks.

4.1 Initialisation of Network Weights

In back-propagation, as in any nonlinear optimization problem, the initialization of the weights has an important influence on the ability of the training algorithm to converge and the speed of that convergence. The error surface is usually very complex, containing many local minima. Since the network weights are normally initialised randomly, a large number of networks with different weight initialisations may need to be trained before a weight initialisation is found which converges to a good solution. If the system generating the data is linear, a neural network will try to fit a nonlinear model to the data. This is an example of overfitting, as the network also tries to model the noise on the data. Even for good prediction of the training data, the performance will deteriorate for independent test data produced by the same system. For input and output data with Gaussian distributions, a linear system minimizing the mean squared prediction error already gives the best parameter estimate for all classes of models, and a nonlinear model cannot improve the performance.

Since the hidden nodes of the network operate predominantly in the linear regions, initialisation of the network to an equivalent linear model may provide a good initialisation with a more rapid convergence of the weights to a useful solution, and may prevent a fruitless search for a better solution if the data is linear. For a general feed-forward network with n_h hidden nodes, no bias term and both delayed input and output samples in the input vector, the predicted output is given by

$$\hat{z}(t) = \sum_{k=1}^{n_h} u_k f\left(\sum_{i=1}^{n_u} v_{k(i+n_z)} u(t-i) + \sum_{j=1}^{n_z} v_{kj} z(t-j)\right) \tag{9}$$

where u_k and v_{ki} are the elements of the weight matrices U and V respectively, and $u(t)$ is the input. Linearisation of (9) gives a weighted sum of n_h parallel ARX models, one for each hidden node. Rearranging the linearized equation gives

$$\hat{z}(t) = \sum_{i=1}^{n_u} (\sum_{k=1}^{n_h} u_k v_{ki}) u(t-i) - \sum_{j=1}^{n_z} (-\sum_{k=1}^{n_h} u_k v_{kj}) z(t-j) \tag{10}$$

This is equivalent to the single ARX model of (8), with

$$b_i = \sum_{k=1}^{n_h} u_k v_{k(i+n_z)} \qquad i = 1 \ldots n_b,\ n_b = n_u \tag{11}$$

$$a_j = -\sum_{k=1}^{n_h} u_k v_{kj} \qquad j = 1 \ldots n_a,\ n_a = n_z \tag{12}$$

Hence the parameters of our equivalent linear model are weighted sums of the weights for each hidden node. There are many alternative factorizations for a_j and b_i, depending on the choice of u_k. Useful factorizations are those which keep the elements of V low in order to prevent saturation of the hidden nodes at the onset of training. Initialisation to a linear model is therefore only of limited use for reducing the number of different weight initializations required to find one which converges to a good solution. Linear initialization cannot guarantee improved convergence and relies on the existence of a reasonable linear model of the data.

For a feed-forward network with $n_h = 1$, $n_u = 0$ and no bias term, the linearisation of (9) becomes a weighted sum of parallel AR models. This is equivalent to a single AR model, given by (8) with $b_i = 0$ for all i. We tested this AR initialisation on the simulation of the sunspot data of section 3.2. We trained several feed-forward networks with $n_h = 1$, $n_u = 0$, $n_i = n_z = 9$ and no bias term, using different random weight initialisations to calculate an average performance. We estimated a linear AR model by the 'Burg' algorithm, as in section 3.2, and used these parameters to initialise the weights of the feed-forward network, taking $u_1 = 1$. We then trained this network and compared the performance with that for random initialisation. It was found that the average prediction gain was 7.48 dB for the random initialisations and 8.03 dB for the linear initialisation, after the same number of training cycles.

4.2 Comparison of Parameter Estimation by Feed-forward and Recurrent Networks

Before investigating the parameter estimation of recurrent and feed-forward networks, it is useful to review some terminology derived from the system identification literature (Ljung, 1987), which describes classes of models for linear time invariant systems.

For a linear time invariant system with input $u(t)$ and output $z(t)$, the simplest input-output relationship is a linear difference equation

$$z(t) + a_1 z(t-1) + \ldots + a_{n_a} z(t-n_a) = b_1 u(t-1) + \ldots + b_{n_b} u(t-n_b) + e(t) \tag{13}$$

This is an equation error model since the white noise term, $e(t)$, appears directly in the difference equation and is considered as an error term that cannot be accounted for by the model structure. For a linear relationship between the input, $u(t)$, and the undisturbed output, $w(t)$

$$w(t) + f_1 w(t-1) + \ldots + f_{n_f} w(t-n_f) = b_1 u(t-1) + \ldots + b_{n_b} u(t-n_b) \tag{14}$$

the actual observed output, $z(t)$, is given by

$$z(t) \quad = \quad w(t) + e(t) \tag{15}$$

This is an output error model since the white noise term, $e(t)$, is considered as white measurement noise at the output. Using these models to perform one-step ahead prediction yields the predictors

$$\hat{z}(t) \quad = \quad \sum_{i=1}^{n_b} b_i u(t-i) - \sum_{j=1}^{n_a} a_j z(t-j) \qquad \text{Equation Error} \tag{16}$$

$$\hat{z}(t) \quad = \quad \sum_{i=1}^{n_b} b_i u(t-i) - \sum_{j=1}^{n_f} f_j \hat{z}(t-j) \qquad \text{Output Error} \tag{17}$$

Estimation of the parameters a, b, f, by the prediction error approach requires the minimisation of a function, E, of the prediction error, $e(t) = z(t) - \hat{z}(t)$. The parameter estimation process is termed *equation error minimisation* and *output error minimisation* for the equation error and output error models respectively. The most common approach is 'nonlinear least squares', in which the mean squared prediction error given by (3) is minimised by gradient-based algorithms, such as the Newton Algorithm, pages 282–284(Ljung, 1987).

The ARX model of (8) is identical to the equation error model of (16), hence performs parameter estimation by equation error minimisation. The AR time series model, with all $b_j = 0$ in (8), is also an equation error model. We have already shown in section 4.1 that when operating predominantly linearly, a single hidden layer feed-forward network is equivalent to an ARX or AR model. Since the estimation of the weights by back-propagation is also a minimisation of the mean squared error, the feed-forward network performs parameter estimation by equation error minimisation.

Replacing $z(t-j)$ by $\hat{z}(t-j)$ in (9), we convert the feed-forward network into a recurrent (or feedback) network. Comparison with (17), with $n_b = n_u$, $n_f = n_z$, shows this to be equivalent to a weighted sum of n_h output error models. By analogy with (10), this is equivalent to the single output error model of (17) for the parameter equivalences given by (11) and (12), with f_j replacing a_j and $n_a = n_f = n_z$. Hence a recurrent network with feedback from the network output to the network input performs parameter estimation by output error minimisation. For systems that measure their performance on the output error, for example speech coding systems, a recurrent network model of the system can give improved performance over a linear or feed-forward nework model, since the output error is minimised directly by estimation of the network parameters. Recurrent networks have successfully been applied to CELP-based speech coders (Wu and Fallside, 1992) to give improved performance over equation error-based models.

The recurrent network structures of section 3.3 have unit delay feedback around the hidden nodes, Fig. 1. When operating predominantly linearly, we see from (1) that each hidden node takes the form of an output error model, but is of a more complex structure than (17) due to the full connectivity of W. The overall network has an output error form and can be considered as a dynamical system in which the hidden node outputs, $Y(t)$, represent the state variables. The inclusion of state variables, or memory, accounts for the observation that recurrent networks are better able to model the dynamics of a system than feed-forward networks (Back and Tsoi, 1991b). For applications in which contextual information is important, for example in speech prediction, recurrent networks can therefore outperform feed-forward networks (Tishby, 1990).

5 Conclusion

For the identification of a system with continuous, real-valued input and output data using a single hidden layer neural network model, it has been found that the hidden nodes operate predominantly in the linear region of their nonlinear function. As the number of hidden nodes increases, saturation may be tolerated without loss of the continuous nature of the network output, provided some of the nodes remain linear. Excursions into slightly nonlinear regions of the hidden node function introduce nonlinearity into the model, which allows generation of limit cycles for the data system (if these exist). This is similar to the building of an electronic oscillator, in which nonlinearity is required to sustain oscillation. If the period of the generated limit cycle is similar to that of the system, the model captures the dynamics of the underlying system that generates the data. This can improve the prediction performance of the model on unseen data from the same system. Therefore, we do gain by introducing some nonlinearity although the network operates predominantly linearly. Generation of poor limit cycles suggests that the network structure is a poor match to that of the underlying system, since the inherent dynamic properties are different.

Since the network is predominantly linear, we can exploit the well-established theories of system identification, which is traditionally based on the estimation of the coefficients of linear models using variants of least squares or maximum likelihood algorithms. These theories provide alternative learning strategies to back-propagation, such as second order gradient techniques. Using system identification methods for model validation, parameter bias estimation, generalization and prediction enables us to study and evaluate the performance of neural networks in a rigorous manner. A linear model can provide a useful solution to the initialisation of the weights of an equivalent neural network model. Although there are many possible factorisations of the equivalent linear model, these factorisations may reduce the number of weight initialisations required to locate one which converges to a good solution. Such a linear initialisation does not avoid the problem of local minima, but could prevent a fruitless search for a better solution if the data is actually linear.

For the linear regions of operation, comparison with the standard linear models of system identification shows that a feed-forward network is equivalent to an AR or ARX model of the data and performs parameter estimation by equation error minimisation. A recurrent network is equivalent to an output error model, performing parameter estimation by output error minimisation, or a dynamical system, in which the output of the hidden nodes may be interpreted as state variables. Hence recurrent networks combine the advantages of nonlinearity and memory to give improved modelling of the dynamical properties of data over both linear models and feed-forward networks.

References

Back, A. D. and Tsoi, A. C. (1991a). Analysis of hidden layer weights in a dynamic locally recurrent network. In Kohonen, T., Makisara, K., Simula, O., and Kangas, J., editors, *Artificial Neural Networks*, volume 1, pages 961–966, Elsevier Science Publishers B. V. (North Holland).

Back, A. D. and Tsoi, A. C. (1991b). FIR and IIR synapses, a new neural network architecture for time series modelling. *Neural Computation*, 3:375–385.

Billings, S., Jamaluddin, H., and Chen, S. (1992). Properties of neural networks with applications to modelling nonlinear dynamical systems. *Int. J. Control*, 55(1):193–224.

Bourland, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biol.Cybern.*, 59:291–294.

Chen, S., Billings, S., and Grant, P. (1990). Nonlinear system identification using neural networks. *Int. J. Control*, 51(6):1191–1214.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Math. Contr. Signals. Syst.*, 2:303–314.

Gallinari, P., Thiria, S., and Fogelman Soulie, F. (1988). Multilayer perceptrons and data analysis. *IEEE Annual Conference on Neural Networks*, 1:391–399.

Garofolo, J. (1988). *Getting Started with the DARPA TIMIT CD-ROM: An Acoustic Phonetic Continuous Speech Database*. National Institute of Standards and Technology (NIST), Gaithersburgh, M.D.

Lapedes, A. and Farber, R. (1987). Nonlinear signal processing using neural networks: prediction and system modelling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory.

Ljung, L. (1987). *System Identification : Theory for the User*. Prentice-Hall, Englewood Cliffs, N.J.

Ljung, L. and Sjöberg, J. (1992). A system identification perspective on neural nets. In Kung, S., editor, *Proc. IEEE Workshop on Neural Networks for Signal Processing*, IEEE Science Center, N.J.

Narendra, K. and Parthasarathy, K. (1990). Identification and control for dynamic systems using neural networks. *IEEE Trans. Neural Networks*, 1:4–27.

Priestley, M. (1988). *Nonlinear and nonstationary time series analysis*. Academic Press.

Schafer, R. W. and Rabiner, L. R. (1978). *Digital Processing of Speech Signals*. Prentice-Hall, Englewood Cliffs, N.J.

Tishby, N. (1990). A dynamical systems approach to speech processing. In *Proc. Int. Conf. on Acoustics, Speech, Signal Processing*, pages 365–368.

Townshend, B. (1991). Nonlinear prediction of speech. In *Proc. Int. Conf. on Acoustics, Speech, Signal Processing*, pages 425–428.

Webb, A. and Lowe, D. (1990). The optimised internal representation of multilayer classifier networks performs linear discriminant analysis. *Neural Networks*, 3:367–375.

White, H. (1989). Learning in artificial neural networks : A statistical perspective. *Neural Computation*, 1:425–464.

Wu, L. and Fallside, F. (1992). Fully vector quantized neural network-based code-excited nonlinear predictive speech coding. Technical Report CUED/F-INFENG/TR.96, Cambridge University, England.