Recurrent Neural Networks and Robust Time Series Prediction

Jerome T. Connor, R. Douglas Martin, *Member, IEEE,* and L. E. Atlas, *Member IEEE*

***Abstract*—We propose a robust learning algorithm and apply it to recurrent neural networks. This algorithm is based on filtering outliers from the data and then estimating parameters from the filtered data. The filtering removes outliers from both the target function and the inputs of the neural network. The filtering is *soft* in that some outliers are neither completely rejected nor accepted.**

To show the need for robust recurrent networks, we compare the predictive ability of least squares estimated recurrent networks on synthetic data and on the Puget Power Electric Demand time series. These investigations result in a class of recurrent neural networks, NARMA(p, q), which show advantages over feedforward neural networks for time series with a moving average component.

Conventional least squares methods of fitting NARMA(p,q) neural network models are shown to suffer a lack of robustness towards outliers. This sensitivity to outliers is demonstrated on both the synthetic and real data sets. Filtering the Puget Power Electric Demand time series is shown to automatically remove the outliers due to holidays. Neural networks trained on filtered data are then shown to give better predictions than neural networks trained on unfiltered time series.

I. Introduction

WE PROPOSE a robust learning algorithm and apply it to recurrent neural networks. This algorithm is based on filtering outliers from the data and then estimating parameters from the filtered data. To show the need for recurrent neural networks, we compare the predictive ability of recurrent networks on synthetic and real data. Conventional least squares methods of fitting neural networks, both recurrent and feedforward, are shown to suffer lack of robustness towards outliers. This sensitivity to outliers demonstrates the need for robust learning algorithms.

Neural network models are usually analyzed from the point of view of nonlinear modeling. To differentiate between feedforward and recurrent neural networks, this paper compares nonlinear AR and linear autoregressive moving average (ARMA) modeling by feedforward and recurrent networks respectively. Section II reviews linear autoregressive moving average (ARMA) models as a point of departure for the use of feedforward and recurrent networks. Section III investigates nonlinear autoregressive moving average (NARMA) models of time series. In particular, neural networks are analyzed as nonlinear extensions of traditional linear models. From this, we see recurrent networks have an advantage over feedforward neural networks in much the same way that ARMA models have advantages over autoregressive models for some types of time series. Section IV examines the predictive ability of recurrent NARMA and feedforward NAR models of various time series.

Section V shows that standard least mean squares estimation techniques lead to neural network models that are not robust to outliers. Section VI proposes a new robust estimation algorithm for neural network parameter estimation. In Section VII, a comparison of recurrent networks and other models on data from a competition in electric load forecasting sponsored by the Puget Sound Power and Light Company is discussed. On the competition data, a recurrent network model gives superior results to feedforward networks and various types of linear models. Last, in Section VIII, recurrent neural networks trained with the robust learning algorithm are demonstrated that lead to even better results for electric load forecasting.

Jerome T. Connor is with Bellcore, Morristown, NJ 07960–6438.

R. Douglas Martin and L. E. Atlas is with the Department of Electrical Engineering, the University of Washington, Lynnwood, WA 98036–7138.

IEEE Log Number 9400177.

II. Linear ARMA Model and Optimal Predictors

The statistical approach to forecasting involves the construction of stochastic models to predict the value of an observation x_t using previous observations. This is often accomplished using linear stochastic difference equation models with random inputs. By far the most important class of such models is the linear autoregressive moving average (ARMA) models. Here we very briefly review linear ARMA models and optimal predictors for these models. Detailed discussions of these models may be found, (e.g., [3], [14]).

A. Linear ARMA(p, q) Models

A very general class of linear models used for forecasting purposes is the class of ARMA(p, q) models

$$x_t = \gamma + \sum_{i=1}^{p} \varphi_i x_{t-i} + \sum_{j=1}^{q} \theta_j e_{t-j} + e_t \qquad (1)$$

where it is assumed that $\mathrm{E}\ (e_t | x_{t-1}, x_{t-2}, \cdots) = 0$. This condition is satisfied for example when the e_t are zero mean, independent and identically distributed, and are independent of past x_t's. It is assumed throughout that e_t has a finite variance σ^2. For a zero mean process x_t the intercept γ is zero. To simplify notation we assume $\gamma = 0$, which is equivalent to replacing x_t by $x_t - \mathrm{E}(x_t)$.

B. Optimum Prediction

The theory of prediction focuses on optimum prediction in the minimum mean squared error sense. It is well known that

the minimum mean squared error predictor, given the infinite past, is the conditional mean

$$\hat{x}_t = \mathrm{E}\,(x_t|x_{t-1}, x_{t-2}, \cdots)$$

assuming the conditional mean exists (see [17] or [29]).

In practice one has only the finite past $x_{t-1}, \cdots, x_1$, commencing with the first observation x_1. In this case, the minimum mean squared error predictor is

$$\hat{x}_t = \mathrm{E}\,(x_t|x_{t-1}, x_{t-2}, \cdots, x_1). \tag{2}$$

C. Approximately Optimal ARMA Predictors

It may be shown that for the ARMA(p, q) model (1), the optimal predictor (2) is approximately given by the recursive algorithm

$$\hat{x}_t = \varphi_1 x_{t-1} + \cdots + \varphi_p x_{t-p} + \theta_1 \hat{e}_{t-1} + \cdots + \theta_q \hat{e}_{t-q} \tag{3}$$

where

$$\hat{e}_{t-j} = x_{t-j} - \hat{x}_{t-j} \quad j = 1, 2, \cdots, q. \tag{4}$$

In this case, the following initial conditions are often used:

$$\hat{x}_0 = \hat{x}_{-1} = \cdots = \hat{x}_{-p+1} = \hat{e}_0 = \cdots = \hat{e}_{-q+1} = 0.$$

For improvements on the initial conditions see Box and Jenkins' discussion of "back-forecasting" [3].

D. Optimal AR(p) Predictors

An AR(p) process model is the following special case of an ARMA(p, q) process model:

$$x_t = \varphi_1 x_{t-1} + \cdots + \varphi_p x_{t-p} + e_t. \tag{5}$$

The corresponding optimal predictor is given by

$$\hat{x}_t = \varphi_1 x_{t-1} + \cdots + \varphi_p x_{t-p}. \tag{6}$$

III. Nonlinear ARMA Models and Network Approximations

Many types of nonlinear models have been proposed in the literature, see for example MARS [11], projection pursuit [12], CART [5], autoregressive models [38], bilinear models [37], and Volterra series expansions [6]. Here we focus on feedforward and recurrent neural networks and how they may be used to approximate nonlinear AR and ARMA models, respectively.

A. Nonlinear Autoregressive Models and Feedforward Networks

A natural generalization of the linear AR(p) model to the nonlinear case would be the nonlinear autoregressive (NAR) model

$$x_t = h(x_{t-1}, x_{t-2}, \cdots, x_{t-p}) + e_t$$

where h is an unknown smooth function. As with (1) we assume that $\mathrm{E}\,(e_t|x_{t-1}, x_{t-2}, \cdots) = 0$, and that e_t has finite variance σ^2. Under these conditions the minimum mean square error optimal predictor of x_t given $x_{t-1}, \cdots, x_{t-p}$ is the conditional mean

$$\hat{x}_t = \mathrm{E}\,(x_t|x_{t-1}, \cdots, x_{t-p}) = h(x_{t-1}, \cdots, x_{t-p}) \quad t \geq p+1. \tag{7}$$

This predictor has mean-squared error σ^2.

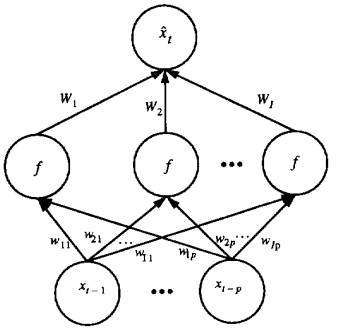


Fig. 1. Feedforward network for NAR(p) modeling.

Feedforward networks were proposed as a NAR model for time series prediction by Lapedes and Farber [21]. A feedforward network is a nonlinear approximation to h given by

$$\hat{x}_t = \hat{h}(x_{t-1}, \cdots, x_{t-p}) = \sum_{i=1}^{I} W_i f\left(\sum_{j=1}^{p} w_{ij} x_{t-j} + \theta_i\right) \tag{8}$$

where the function $f(x)$ is a smooth bounded monotonic function, typically a sigmoid. This architecture is illustrated in Fig. 1, where the sigmoid functions are denoted as circles and the weights are connections between circles.

The parameters W_i and w_{ij} are estimated from a training sample $x_1^0, \cdots, x_N^o$, thereby obtaining an estimate $\hat{h}$ of h. Estimates are obtained by minimizing the sum of the squared residuals $\Sigma_{t=1}^{n}(x_t - \hat{x}_t)^2$. This is done for example by a gradient descent procedure known as "backpropagation" see [35] or by a second order method [2].

More complicated feedforward networks exist than (8). We have avoided the use of multiple layer neural networks because they have not given any advantages for the simple time series problems considered in this paper. In more complicated situations such as speech recognition, multiple layered networks must be considered. For instance, the time-delay neural network (TDNN) of Waibel *et al.* [39] is often used for speech recognition. No matter how complicated the feedforward architecture becomes, such as the TDNN, it is always a member of the class of nonlinear models described in (7) with some finite value of p.

We have also avoided the use of direct linear connections from the input to the output. A single neuron can approximate a linear model by constraining the output to always fall within the linear region of the neuron. In practice, we have found that this often occurs.

B. Nonlinear ARMA Models and Recurrent Networks

This section is devoted to expressing recurrent networks as extensions of traditional ARMA time series models. Recurrent

networks are neural networks that have feedback. Recurrent networks can be used in either continuous or discrete time. The work in this paper is done in discrete time.

Broadly speaking, there are two types of recurrent networks: relaxation and standard. Relaxation networks start from a given state and settle to a fixed point, typically denoting a class. By imposing constraints on the feedback connections, the convergence can be guaranteed. It is possible for relaxation networks to never reach a fixed point, but such behavior is generally considered undesirable because it does not aid in classification. For examples of relaxation networks, see [15], [16], and [33].

We use recurrent networks which try to model a trajectory rather than reach a fixed point. For time series prediction, the changing input results in a recurrent neural network that usually never reaches a fixed point. We use recurrent networks that can predict a continuous range of values, differing from relaxation networks which are used for classification. For a review of recurrent networks see [10], [18], [40], and [41]. As is shown in Section III-C, the recurrent network introduced in this section can be viewed as a special case of that discussed in [40]. For a review on the various topologies of recurrent network see [30].

One natural nonlinear generalization of the linear ARMA model to the nonlinear case is given by

$$x_t = h\left(x_{t-1}, x_{t-2}, \cdots, x_{t-p}, e_{t-1}, \cdots, e_{t-q}\right) + e_t \qquad (9)$$

where h is an unknown smooth function, and as in (1) it is assumed that $\mathrm{E}\left(e_t | x_{t-1}, x_{t-2}, \cdots\right) = 0$ and that the variance of e_t is σ^2. We call this a NARMA(p, q) model. In this case, the conditional mean predictor based on the infinite past of observations is

$$\hat{x}_t = \mathrm{E}\left[h(x_{t-1}, \cdots, x_{t-p}, e_{t-1}, \cdots, e_{t-q}) | x_{t-1}, x_{t-2}, \cdots\right].$$

Suppose that the NARMA model is invertible in the sense that there exists a function g such that

$$x_t = g(x_{t-1}, x_{t-2}, \cdots) + e_t.$$

Then given the infinite past of observations $x_{t-1}, x_{t-2}, \cdots$, one can in principle use the above equation to computed the e_{t-j} in (9) as a function of x_u:

$$e_{t-j} = e_{t-j}(x_{t-j}, x_{t-j-1}, \cdots) \quad j = 1, \cdots, q. \qquad (10)$$

In this case the conditional mean estimate is

$$\hat{x}_t = h\left(x_{t-1}, \cdots, x_{t-p}, e_{t-1}, \cdots, e_{t-q}\right) \qquad (11)$$

where the e_{t-j} are specified by (10) in terms of present and past (relative to time $t-j$) x_u. The predictor (11) has a mean-squared error σ^2.

Since in practice one has only a finite observation record, one can not compute (10) and (11). However, by analogy with the recursive computation of the predictor $\hat{x}_t$ for the linear ARMA process, given by (3) and (4), it seems reasonable to approximate the conditional mean predictor (11) by the recursive algorithm

$$\hat{x}_t = h\left(x_{t-1}, x_{t-2}, \cdots, x_{t-p}, \hat{e}_{t-1}, \hat{e}_{t-2}, \cdots, \hat{e}_{t-q}\right) \qquad (12)$$

$$\hat{e}_j = x_j - \hat{x}_j, \quad j = t-1, \cdots, t-q \qquad (13)$$

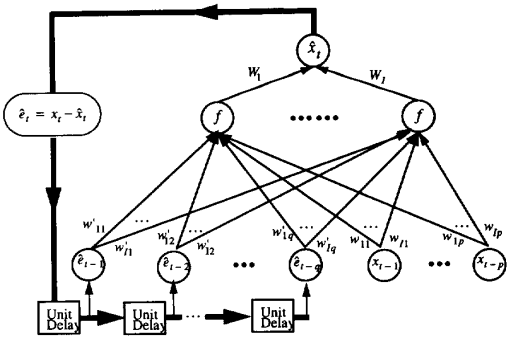


Fig. 2. Recurrent NARMA(p,q) modeling.

with appropriate initial conditions [e.g., those following (4)].

The approximate conditional mean predictor model (12), (13) can be approximated by the following NARMA(p,q) recurrent network model:

$$\hat{x}_t = \sum_{i=1}^{I} W_i f\left(\sum_{j=1}^{p} w_{ij} x_{t-j} + \sum_{j=1}^{q} w'_{ij}(x_{t-j} - \hat{x}_{t-j}) + \theta_i\right). \qquad (14)$$

This recurrent network topology is shown in Fig. 2. The parameters W_i, w_{ij}, and w'_{ij} are estimated by *least squares*, just as in the case of the feedforward network, i.e., by choosing the above parameters to minimize $\Sigma\left(x_t - \hat{x}_t\right)^2$.

C. Fully Interconnected Network

The NARMA recurrent network (14) is a special case of the somewhat more general recurrent network, shown in Fig. 3. For this network

$$\hat{x}_t = \sum_{i=1}^{I} W_i g_i(t) \qquad (15)$$

where the I hidden outputs $g_i(t)$ are computed recursively in time as follows:

$$g_i(t) = f\left(\sum_{j=1}^{\max(p,q)} \tilde{w}_{ij} x_{t-j} + \sum_{k=1}^{q}\sum_{l=1}^{I} \tilde{w}_{ilk} g_l(t-k) + \theta_i\right). \qquad (16)$$

The weights, $\tilde{w}_{ilk}$ are distinct from the doubly subscripted weight $\tilde{w}_{ij}$.

We can now see that (14) is a special case of (15) and (16) as follows:

First set:

$$\tilde{w}_{ilk} = -\tilde{w}'_{ik} W_l \quad \begin{matrix} 1 \le i, l \le I \\ 1 \le k \le q \end{matrix} \qquad (17)$$

and then for the case where $p > q$

$$\tilde{w}_{ij} = \begin{cases} w_{ij} + w'_{ij} & 1 \le j \le q \\ w_{ij} & q+1 \le j \le p. \end{cases} \qquad (18)$$

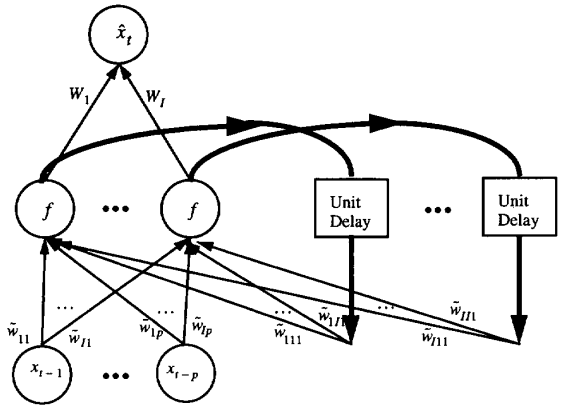


Fig. 3. Fully recurrent network.

This claim is verified by direct substitution of (17) and (18) into (16):

$$g_i(t) = f\left(\sum_{j=1}^{p} \tilde{w}_{ij} x_{t-j} - \sum_{k=1}^{q} w'_{ik} \sum_{l=1}^{I} W_l g_l(t-k) + \theta_i\right) \tag{19}$$

$$= f\left(\sum_{j=1}^{q} (w_{ij} + w'_{ij}) x_{t-j} + \sum_{m=q+1}^{p} w_{im} x_{t-m} - \sum_{k=1}^{q} w'_{ik} \hat{x}_{t-k} + \theta_i\right) \tag{20}$$

$$= f\left(\sum_{j=1}^{p} w_{ij} x_{t-j} - \sum_{k=1}^{q} w'_{ik} (x_{t-k} - \hat{x}_{t-k}) + \theta_i\right). \tag{21}$$

The weight relationships (17) and (18) are modified in an obvious way for the cases $p = q$ and $p < q$.

To distinguish the recurrent network (RN) given by (14), from the more general neural network (15), (16), we refer to (15) and (16) as a NARMA *fully recurrent network* (FRN). Because there may be some potential for the richer class of NARMA FRN to provide a better approximation to $h(x_{t-1}, \cdots, x_{t-p}, e_{t-1}, \cdots, e_{t-q})$ than the NARMA RN class, we consider both of these classes of approximations in the remainder of the paper. Bear in mind however that the NARMA FRN class can require many more weights/parameters than the NARMA RN class. We compute the parameter estimates for both RN and FRN architectures by variations of real-time recurrent learning (RTRL), see [34].

IV. Fitting Feedforward and Recurrent Networks to Simulated Data

From the previous sections, one sees that a recurrent network should have no advantage over a feedforward network in modeling and predicting nonlinear autoregressive models, but that a recurrent network could have considerable advantages for modeling and predicting NARMA processes. To confirm this, feedforward, NARMA recurrent RN, and Fully Recurrent FRN neural networks given by [8], and [14]–[16], respectively, were trained and tested on linear AR(1) and MA(1) processes, and on a Bilinear NARMA(1,1) process. The stopping criteria and the number of hidden units of the neural network model was chosen by way of cross validation. The mean squared errors of Tables I–III are normalized so that the best infinite past predictor would have a normalized mean squared error of 1. We also give standard errors of the estimated mean square error in parenthesis. Note that the small number of hidden units in the following models is not due to the complexity of the learning algorithm but rather to the better generalization of parsimonious models.

TABLE I
Neural Network Performance on AR(1) Time Series.

Network	p	q	I	Training Set M.S.E.	Testing Set M.S.E.
Feedforward NAR	1	0	3	1.00	1.02 (.020)
	2	0	3	1.03	1.08 (.021)
	3	0	2	1.00	1.03 (.020)
	4	0	3	.99	1.06 (.021)
	5	0	3	.98	1.09 (.021)
Recurrent NARMA	0	1	4	2.03	2.12 (.043)
	1	1	3	.99	1.01 (.020)
Fully Recurrent	1	–	5	1.01	1.04 (.020)

A. Gaussian AR(1) Process

The following Gaussian AR(1) process was generated

$$x_t = 0.9 x_{t-1} + e_t,$$

where the e_t are N (0, 1) white noise. The parameters of the network were estimated using the first 500 observations of a time series generated from the AR(1) model. The neural network model was tested on an independent AR(1) time series of 10 000 observations. The results are shown in Table I.

The testing set error is the basis of model comparison. Training set error is indicative of whether the model has overfitted the data. In this example, the MSE on the training set should be at best equal to 1. For many of the networks the MSE less than 1 indicates that some of the noise were modeled, which is not good from a prediction perspective. For example, the NAR(5) model has a lower error on the training set, but relatively poor performance on the testing set.

All the networks model the AR(1) time series well with the exception of the Recurrent NARMA(0, 1) network, which is the only network denied the autoregressive part. The testing set MSE for NAR(1) is close to the theoretical best predictor, which indicates that nonlinear neural network models can do a good job of modeling linear Gaussian (AR) processes.

B. Gaussian MA(1) Process

A Gaussian MA(1) process is given by

$$x_t = e_t + 0.9 e_{t-1}$$

TABLE II
NEURAL NETWORK PERFORMANCE ON MA(1) TIME SERIES.

Network	p	q	I	Training Set M.S.E.	Testing Set M.S.E.
Feedforward NAR	1	0	6	1.30	1.36 (.027)
	2	0	5	1.25	1.25 (.024)
	3	0	3	1.11	1.16 (.023)
	4	0	3	1.08	1.14 (.023)
	5	0	3	1.05	1.09 (.022)
Recurrent NARMA	0	1	3	1.00	1.02 (.020)
	1	1	4	0.99	1.02 (.020)
Fully Recurrent	1	–	4	0.99	1.05 (.021)

TABLE III
NEURAL NETWORK PERFORMANCE ON A BILINEAR NARMA(1,1) TIME SERIES.

Network	p	q	I	Training Set M.S.E.	Testing Set M.S.E.
Feedforward NAR	1	0	7	1.17	1.48 (.028)
	2	0	8	1.11	1.35 (.026)
	3	0	6	1.15	1.50 (.030)
	4	0	4	1.14	1.45 (.028)
	5	0	4	1.07	1.50 (.029)
Recurrent NARMA	0	1	2	1.49	1.79 (.035)
	1	1	5	1.11	1.28 (.025)
Fully Recurrent	1	–	5	1.03	1.11 (.022)

where the e_t are N(0,1) white noise. The parameters of the network were estimated using the first 500 observations of a time series generated from the MA(1) model. The neural network model is tested on a time series of 10 000 observations. The results are shown in Table II.

As should have been expected, the recurrent NARMA network models the time series better than all feedforward networks. The testing set error for the NARMA network predictors is close to the theoretical best MA(1) model predictor, which indicates that nonlinear neural network models can model linear Gaussian MA processes.

Unlike the NAR(1) example, the greater order of the feedforward NAR(p) model leads to lower training and testing set errors for the MA(1) time series. For the NAR(p) model, the cost in complexity of large p is countered by the availability of useful data. Finally, it should be noticed that as number of past observations incorporated in the feedforward network is increased, the testing set error approaches the theoretical limit: i.e., high order AR models approximate low order MA models.

C. Bilinear NARMA(1,1) Process

The following bilinear NARMA(1,1) process

$$x_t = e_t + 0.5e_{t-1}x_{t-1}$$

was generated, where the e_t are N (0, 1) white noise. The parameters of the network are determined by the first 500 observations of the time series generated from the bilinear NARMA(1,1) model. The results are shown in Table III.

The recurrent NARMA(0,1) model performs poorly due to the lack of an autoregressive component which is needed in prediction of the bilinear NARMA(1,1) time series. Both the recurrent NARMA(1,1) and the fully recurrent networks outperform all other networks (including all feedforward NAR networks) on the bilinear NARMA(1, 1) time series. The good performance of the fully recurrent network relative to the recurrent NARMA networks indicates that the fully recurrent network may be better at modeling some nonlinearities.

Adding previous observations to the feedforward NAR does improve performance, but only up to $p = 2$. The performance for $p = 5$ has the second lowest error on the training set, but has the second highest error rate on the testing set. Again, greater parsimony and better input representation lead to better generalization.

V. NON-ROBUSTNESS OF LS METHODS FOR NEURAL NETWORKS

Both feedforward nonlinear autoregression (NAR) and recurrent nonlinear autoregression moving average (NARMA) models are traditionally fit by a least squares (LS) criterion. For example in the case of fitting a feedforward NAR(p) type network model, one estimates the model by minimizing the sum of the squared residuals

$$\min_g \sum_{t=p+1}^{n} (y_t - g(\boldsymbol{y}_{t-1}))^2 \tag{22}$$

where $\boldsymbol{y}_{t-1}^{\mathrm{T}} = (y_{t-1}, y_{t-2}, \cdots, y_{t-p+1})$ and

$$g(\boldsymbol{y}_{t-1}) = \sum_{i=1}^{I} W_i \sigma(\boldsymbol{y}_{t-1}^{\mathrm{T}} \boldsymbol{w}_i + \theta_i) \tag{23}$$

with input weight vectors $\boldsymbol{w}_i^{\mathrm{T}} = (w_{i1}, w_{i2}, \cdots, w_{ip})$ for the ith hidden unit, and output weights W_i, for each hidden unit $i = 1, 2, \cdots, I$. The minimization of (22), as mentioned in Section III, can be done using any one of various gradient methods; e.g., backpropagation with gradient descent, Gauss-Newton type optimization, etc.

A. Fitting NAR(1) Processes with A NAR(1) Type Neural Network

1) A NAR(1) Model Time Series with a Single Additive Outlier: To observe the effect of a single outlier on least squares fitting of a neural network predictor model, consider the process

$$y_t = x_t + v_t \tag{24}$$

where

$$x_t = 1.5x_{t-1}e^{-(x_{t-1}^2/4)} + e_t = g(x_{t-1}) + e_t \tag{25}$$

and

$$v_t = \begin{cases} 20 & t = 100 \\ 0 & t \neq 100 \end{cases} \tag{26}$$

with the e_t being i.i.d. N(0, 1). Fig. 4 shows a time series of length $n = 200$ generated from (24), (25), and (26).

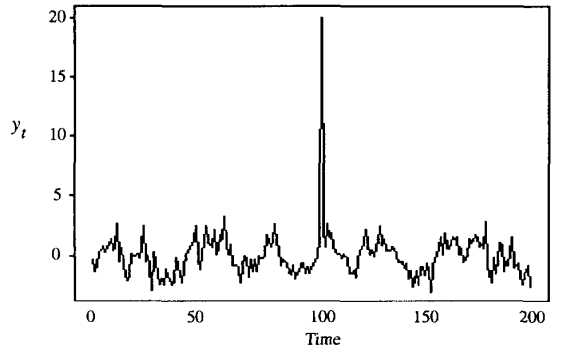


Fig. 4(a). Time series generated by a NAR(1) model.

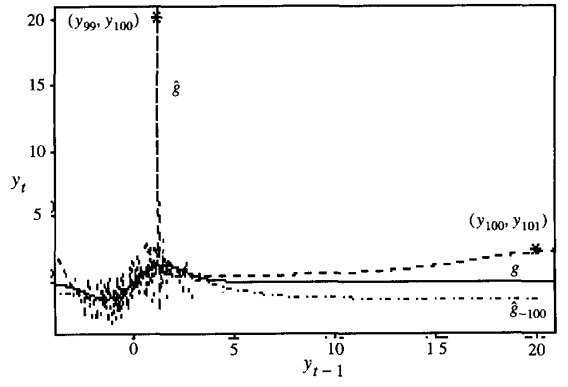


Fig. 4(b). Scatter plot of y_t vs. y_{t-1} for (24) and (26), with true predictor $g$and least squares fitted predictor $\hat{g}$ and $\hat{g}_{100}$.

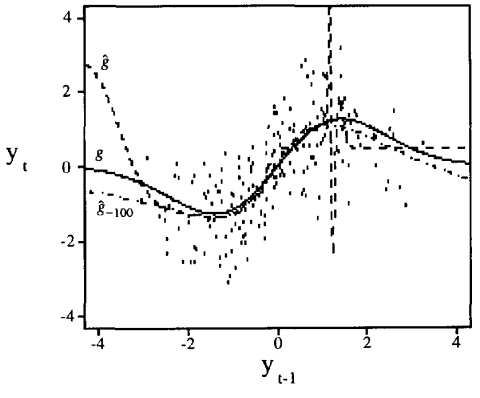


Fig. 4(c). Local view of Fig. 4(b).

Fig. 4(b) displays a scatter plot of y_t versus y_{t-1}, along with the following three superimposed curves:

1) The true model conditional mean (best) predictor

$$g(y_{t-1}) = 1.5y_{t-1}e^{-(y_{t-1}^2/4)} \tag{27}$$

assuming that all v_t's are zero.

2) The least squares estimated conditional mean NAR(1) predictor $\hat{g}$ based on the y_t as given by (24)–(26),

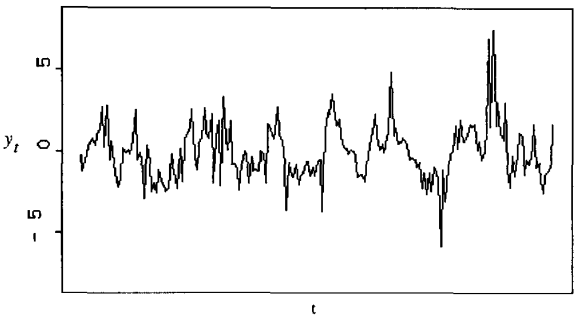


Fig. 5(a). Time series for NAR(1) model (24), (26), and (30) with 10% outliers.

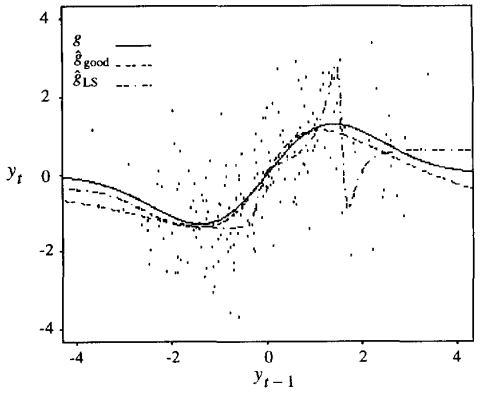


Fig. 5(b). Lag one, scatter plot for 10% outlier data with the true, and least squares and robust NAR(1) estimates.

namely

$$\hat{g}(y_{t-1}) = \sum_{i=1}^{10} \hat{W}_i \sigma(y_{t-1}\hat{w}_i + \hat{\theta}_i). \tag{28}$$

3) The least squares estimated conditional mean NAR(1) predictor $\hat{g}_{\text{good}}$ based on the outlier free x_ts: i.e., based on deleting the single additive outlier at $t = 100$:

$$\hat{g}_{\text{good}}(y_{t-1}) = \sum_{i=1}^{4} \hat{W}_i \sigma(y_{t-1}\hat{w}_i + \hat{\theta}_i). \tag{29}$$

The effect of a single outlier on the NAR(1) least squares neural network fit is clearly shown in Fig. 4(b). The first outlier data point (y_{99}, y_{100}) has a large local effect on the fitted predictor model, while the other outlier data point (y_{100}, y_{101}) produces a semi-global effect on the predictor when y_{t-1} is large and positive (the fit is not very good for large negative values of y_{t-1}). Fig. 4(c) is a local view of Fig. 4(b), which shows more clearly the local impact of the outlier in the vicinity of $y_{t-1} = 1.3$. In summary: the neural network model fitted by least squares does a reasonable job for a wide range of predictor values of y_{t-1}, but fails rather badly in local regions of y_{t-1} and for large values of $|y_{t-1}|$.

2) A NAR(1) Model Time Series with Multiple Additive Outliers: Fig. 5(a) shows a time series of length $n = 200$ generated by the model (24) and (26), in the preceding NAR(1)

single outlier example. Instead of the single additive outlier described in (25), multiple additive outliers are allowed. The ε_t are white Gaussian noise with mean zero and variance $\sigma_\varepsilon^2 = 1$. The v_t are generated in the following way

$$v_t = W_t \cdot 3Z_t \tag{30}$$

where the Z_t are independent standard normal random variables, and the W_T are independent 0-1 (Bernoulli) random variables with $P(W_t = 1) = \gamma$. Thus, γ is the average fraction of outliers, whose values are $3Z_t \sim N(0, 9)$, where the notation $N(\mu, \sigma^2)$ stands for the normal distribution with mean μ and variance σ^2.

Fig. 5(a) is for $\gamma = 0.1$: i.e., 10% outliers on average (with 12% outliers for this realization). Fig. 5(b) displays the lag-one scatterplots for the series of Fig. 5(a) along with:

1) The true predictor g (solid line) given by equation (27).
2) The least squares estimated conditional mean NAR(1) predictor $\hat{g}$ based on the y_t as given by (24), (26) and (30) with $\gamma = 0.1$ corresponding to 10% outliers, namely

$$\hat{g}(y_{t-1}) = \sum_{i=1}^{5} \hat{W}_i \sigma(y_{t-1}\hat{w}_i + \hat{\theta}_i). \tag{31}$$

3) The least squares estimated conditional mean NAR(1) predictor $\hat{g}_{\text{good}}$ based on the outlier free data, i.e., with all $v_t = 0$.

The message is clear: While the least squares estimate $\hat{g}_{\text{good}}$ based on outlier free data is a reasonable good estimate of the true predictor g, the least squares estimate $\hat{g}$ is badly affected by outliers, and would provide very poor predictions for some values y_{t-1}.

VI. Robust Filtering of Feedforward and Recurrent Predictor Models

Previous work in robust estimation of neural network modeling (see [20]), would limit the influence of an outlier when it appears in the target function. Our robust ARMA model fitting method limits the influence of an outlier when it is both the target and an input of the neural network model.

The robust ARMA model fitting method of [24] is applied to the case of feedforward and recurrent networks of NAR and NARMA types, respectively. The method relies on a robust filtering method for interpolating at times of occurrence of outliers, and is related to a non-Gaussian maximum likelihood estimation procedure.

A. Robust Filters For NARMA Models

The NARMA(p, q) model

$$x_t = f(x_{t-1}, \cdots, x_{t-p}, \varepsilon_{t-1}, \cdots, \varepsilon_{t-q}) + \varepsilon_t \tag{32}$$

may be written in $(p + q)$ dimensional vector state space form as follows. Define $(p + q)$ dimensional column vectors $\boldsymbol{x}_t, \boldsymbol{f}(\boldsymbol{x}_{t-1})$ and $\boldsymbol{\varepsilon}_t$ as follows:

$$\boldsymbol{x}_t = (x_t, \cdots, x_{t-p+1}, \varepsilon_t, \cdots, \varepsilon_{t-q+1})^T \tag{33}$$

$$\boldsymbol{f}(\boldsymbol{x}_{t-1}) = [f(\boldsymbol{x}_{t-1}), x_{t-1}, \cdots, x_{t-p+1}, 0, \varepsilon_{t-1}, \cdots, \varepsilon_{t-q+1}]^T \tag{34}$$

$$\boldsymbol{\varepsilon}_t = (\varepsilon_t, 0, \cdots, 0, \varepsilon_t, 0, \cdots 0)^T \tag{35}$$

where the second ε_t in (35) occurs at the $(p + 1)$th position.

Then the NARMA(p, q) model with additive outliers may be written as

$$\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_{t-1}) + \boldsymbol{\varepsilon}_t \tag{36}$$

$$y_t = \boldsymbol{h}^{\mathrm{T}}\boldsymbol{x}_t + v_t \tag{37}$$

where v_t represents additive outliers and $\boldsymbol{h} = (1, 0, 0, \cdots, 0)^{\mathrm{T}}$ is a $p + q$ dimensional vector. Let Q denote the covariance matrix of $\boldsymbol{\varepsilon}_t$.

Let $\hat{\boldsymbol{x}}_t$ denote the robust filter esimate of $\boldsymbol{x}_t$ based on y_1 up to y_t and let $\hat{\boldsymbol{x}}_t^{t-1}$ denote the robust one step ahead predictor of $\hat{\boldsymbol{x}}_t$ based on observations y_1 up to y_{t-1}. Similarly let $\hat{x}_t^{t-1}$ and $\hat{y}_t^{t-1}$ be the one step ahead predictor x_t and y_t respectively, based on y_1 up to y_{t-1}.

The form of robust filter for this model is similar to those proposed by [20], [25], and [28] for linear state space models. The main differences consists of mimicking the so-called "extended" Kalman filter for nonlinear state space models, by making use of linearization of $\boldsymbol{f}(\boldsymbol{x}_{t-1})$ about the robust filter estimate $\hat{\boldsymbol{x}}_{t-1}$:

$$\boldsymbol{f}(\boldsymbol{x}_{t-1}) \approx \boldsymbol{f}(\hat{\boldsymbol{x}}_{t-1}) + \boldsymbol{D}\boldsymbol{f}(\hat{\boldsymbol{x}}_{t-1})(\boldsymbol{x}_{t-1} - \hat{\boldsymbol{x}}_{t-1}) \tag{38}$$

where $\boldsymbol{D}\boldsymbol{f}(\hat{\boldsymbol{x}}_{t-1})$ is the $(p+q) \times (p+q)$ matrix of the partial derivatives of $\boldsymbol{f}$ evaluated at $\hat{\boldsymbol{x}}_{t-1}$. Specifically,

$$\boldsymbol{D}\boldsymbol{f}(\hat{\boldsymbol{x}}_{t-1}) = \left[\begin{array}{c:c:c} \boldsymbol{F}_1 \boldsymbol{F}_2 \cdots \boldsymbol{F}_p & \boldsymbol{F}_{p+1} & \boldsymbol{F}_{p+2} \boldsymbol{F}_{p+3} \cdots \boldsymbol{F}_{p+q} \\ \hdashline I & 0 & 0 \\ \hdashline 0 & 0 & 0 \\ \hdashline 0 & 0 & I \end{array}\right] \tag{39}$$

where $\boldsymbol{F}_i = (\partial/\partial(\boldsymbol{x}_t)_i) f(\boldsymbol{u})|_{\boldsymbol{u}=\hat{\boldsymbol{x}}_{t-1}}$ for $1 \le i \le p + q$.

The one step ahead state and observation predictions of the NARMA(p, q) process defined in (36) and (37) is given by

$$\hat{\boldsymbol{x}}_t^{t-1} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{t-1}) \tag{40}$$

$$\hat{y}_t^{t-1} = \boldsymbol{h}^{\mathrm{T}}\boldsymbol{f}(\hat{\boldsymbol{x}}_{t-1}) = f(\hat{\boldsymbol{x}}_{t-1}). \tag{41}$$

where the estimated state at time $t - 1$ is given by $\hat{\boldsymbol{x}}_{t-1}$. Also note that (41) is the first element of the vector in (40).

The robust filter recursion estimate of $\hat{\boldsymbol{x}}_{t-1}$ is given by

$$\hat{\boldsymbol{x}}_t = \boldsymbol{f}(\hat{\boldsymbol{x}}_{t-1}) + \frac{M_t \boldsymbol{h} s_t}{s_t^2} \psi\left(\frac{y_t - \hat{y}_t^{t-1}}{s_t}\right) \tag{42}$$

where for robustifying function ψ we use the Hampel two-part redescending function ψ_{HA} shown in Fig. 6. Note that $s_t\psi((y_t - \hat{y}_t^{t-1})/s)$ is the robustly estimated residual. The one step ahead state prediction covariance, M_t, used in (42) is given by

$$M_{t+1} = Df(\hat{\boldsymbol{x}}_t) P_t Df^{\mathrm{T}}(\hat{\boldsymbol{x}}_t) + Q \tag{43}$$

$$P_t = M_t - M_t \boldsymbol{h} w\left(\frac{y_t - \hat{y}_t^{t-1}}{s_t}\right) \tag{44}$$

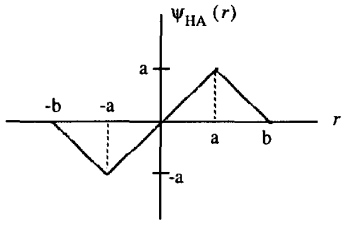


Fig. 6. Hampel psi function.

where $w(r) = (\psi(r)/r)$ and $s_t^2 = (M_t)_{11}$.

In the case of a NARMA(p,q) model we have

$$\hat{\boldsymbol{x}}_{t-1} = (\hat{x}_{t-1}, \cdots, \hat{x}_{t-p}, \hat{\varepsilon}_{t-1}, \cdots, \hat{\varepsilon}_{t-q}) \tag{45}$$

and we need expressions for $\hat{\varepsilon}_{t-1}, \cdots, \hat{\varepsilon}_{t-q}$. These are readily obtained by analogy to (12) and (13).

$$\hat{\varepsilon}_u = \hat{x}_u - f(\hat{\boldsymbol{x}}_{u-1}) = \hat{x}_u + \hat{x}_u^{u-1}, \qquad u = t-1, \cdots, t-q, \tag{50}$$

where x_u has been replaced by $\hat{x}_u$. Note also our change in notation in (12) and (13), $\hat{x}_u$ denoted a one step ahead predictor and now $\hat{x}_u^{u-1}$ denotes a one step ahead predictor and $\hat{x}_u$ denotes a filtered value.

Notice that use of $\psi = \psi_{\mathrm{HA}}$ results in the following behavior for the robust filter:

1) $\hat{x}_t$ is a pure prediction

$$\hat{x}_t = f(\boldsymbol{x}_{t-1})$$

when $|y_t|$ is sufficiently large due to the presence of an additive outlier: i.e., when $|y_t - y_t^{t-1}| > bs_t$, and

2) $\hat{x}_t = (\hat{\boldsymbol{x}}_t)_1 = y_t$ when the magnitude of $|y_t - \hat{y}_t^{t-1}|$ is "small", i.e., when $|y_t - \hat{y}_t^{t-1}| < as_t$. The latter condition holds for the bulk of the observations when no outliers are present.

Results of [24] and [28] suggest that if ψ is the score function of the observation prediction density $f(y_t|\boldsymbol{y}^{t-1})$, then $\hat{\boldsymbol{x}}_t$ and $\hat{\boldsymbol{x}}_t^{t-1}$ are approximate conditional mean estimates (see also [23]). This provides a good rational for the above robust filter recursions.

The above robust filter recursions include NAR(p) and NMA(q) filters as special cases in the following way. The NAR(1) robust filter is given by

$$\hat{x}_t = f(\hat{x}_{t-1}) + s_t\psi\left(\frac{y_t - f(\hat{x}_{t-1})}{s_t}\right) \tag{51}$$

$$m_{t+1} = [f'(\hat{x}_t)]^2 p_t = \sigma_\varepsilon^2 \tag{52}$$

$$p_t = \left[1 - w\left(\frac{y_t - f(\hat{x}_{t-1})}{s_t}\right)\right] m_t. \tag{53}$$

Note the manner in which f influences the state prediction error variance m_{t+1}: high values of $|f'(\hat{x}_t)|$ increase m_{t+1}, and when $|f'(\hat{x}_t)|$ is close to zero we have $m_{t+1} \approx \sigma_\varepsilon^2$.

The NMA(1) robust filter is given by

$$\hat{x}_t = f(\hat{\varepsilon}_{t-1}) + s_t\psi\left(\frac{y_t - f(\hat{\varepsilon}_{t-1})}{s_t}\right), \ \hat{\varepsilon}_t = \hat{x}_t - f(\hat{\varepsilon}_{t-1}) \tag{54}$$

$$m_{t+1} = [f'(\hat{\varepsilon}_{t-1_t})]^2 p_t + \sigma_\varepsilon^2 \tag{55}$$

$$p_t = \left[1 - w\left(\frac{y_t - f(\hat{\varepsilon}_{t-1})}{s_t}\right)\right] m_t. \tag{56}$$

Note once again the manner in which f influences the state prediction error variance m_{t+1}: high values of $|f'(\hat{x}_t)|$ increase m_{t+1}, and when $|f'(\hat{x}_t)|$ is close to zero we have $m_{t+1} \approx \sigma_\varepsilon^2$.

B. Robust Neural Network Training

As in the previous section, let

$$\begin{aligned}\hat{y}_t^{t-1} &= f(\hat{\boldsymbol{x}}_{t-1}) \\ &= f(\hat{x}_{t-1}, \cdots, \hat{x}_{t-p}, \hat{\varepsilon}_{t-1}, \cdots, \hat{\varepsilon}_{t-q})\end{aligned} \tag{57}$$

be the robust one step ahead predictor of y_t, with the $\hat{x}_{t-1}, \cdots, \hat{x}_{t-p}$ computed using the robust NARMA filter recursion of the previous section. But now let f be approximated by a recurrent neural network as in Section III-B:

$$\begin{aligned}f(\hat{\boldsymbol{x}}_{t-1}) &\approx g(\hat{\boldsymbol{x}}_{t-1}) \\ &\quad + \sum_{i=1}^{I} W_i\sigma\left(\sum_{j=1}^{p} w_{ij}\hat{x}_{t-j} + \sum_{j=1}^{q} w'_{ij}\hat{\varepsilon}_{t-j} + \theta_i\right)\end{aligned} \tag{58}$$

with

$$\hat{\varepsilon}_{t-j} = \hat{x}_{t-j} - \hat{x}_{t-j}^{t-j-1} \tag{59}$$

where $\hat{x}_{t-j}^{t-j-1}$ is the one step ahead robust predictor of x_{t-j}. It is important to use the robustly filtered value $\hat{x}_{t-j}$ rather than the observation y_{t-j}, in order to produce cleaned estimates $\hat{\varepsilon}_{t-j}$, which are free of outliers.

Now the robustly filtered values are based on the state transiton equation

$$\hat{\boldsymbol{x}}_t = \boldsymbol{g}(\hat{\boldsymbol{x}}_{t-1}) + \boldsymbol{\varepsilon}_t \tag{60}$$

where $\boldsymbol{g}$ is identical to the $\boldsymbol{f}$ in (34) except that the first component $f(\boldsymbol{x}_{t-1})$ of $\boldsymbol{f}$ is replaced by the approximation $g(\hat{\boldsymbol{x}}_{t-1})$. In the robust NARMA filter equations (40)–(44) g and $\boldsymbol{g}$ replace f and $\boldsymbol{f}$ respectively.

Denote the set of parameters in (58) by $\boldsymbol{\lambda}$

$$\boldsymbol{\lambda} = \{\boldsymbol{W}^{\mathrm{T}}, \boldsymbol{w}_1^{\mathrm{T}}, \cdots, \boldsymbol{w}_I^{\mathrm{T}}, \boldsymbol{w}_1'^{\mathrm{T}}, \cdots, \boldsymbol{w}_I'^{\mathrm{T}}, \theta^{\mathrm{T}}\} \tag{61}$$

and acknowledge the dependence of g on $\boldsymbol{\lambda}$ by writing

$$g(\hat{\boldsymbol{x}}_{t-1}) = g(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda}). \tag{62}$$

It is known ([24], [26]) that an approximate maximum likelihood type estimate (M-estimate) $\hat{\boldsymbol{\lambda}}$ for the model [24] with x_t a Gaussian ARMA model and $P(v_t \neq 0) = \gamma > 0$, is given by

$$\hat{\boldsymbol{\lambda}} = \underset{\boldsymbol{\lambda}}{\operatorname{argmin}} \left\{\sum_{t=1}^{n} \log\, s_t + \sum_{t=1}^{n} \rho\left(\frac{y_t - \hat{y}_t(\boldsymbol{\lambda})}{s_t}\right)\right\}. \tag{63}$$

The two main types of ρ in (63) that have been used in practice are the Huber function ρ_{H} and the Tukey "bisquare" function ρ_{B} shown in Fig. 7(a) and (b). Both of these functions are quadratic for sufficiently small argument r. The Huber

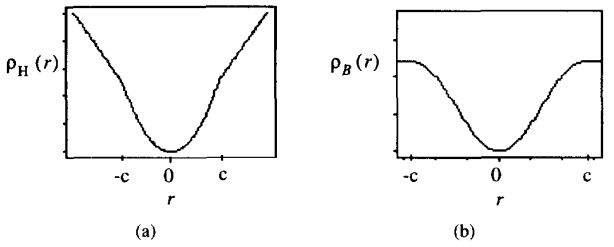


Fig. 7. Huber and Tukey rho functions.

function ρ_{H} is unbounded and convex, with $\rho_{\mathrm{H}}(r)$ behaving like $|r|$ for $|r| \geq c$. The Tukey function ρ_{B} is bounded and hence non-convex.

It has been found in practice that there is usually not much difference between the $\hat{\boldsymbol{\lambda}}$ given by (63) and the following somewhat simpler statement:

$$\hat{\boldsymbol{\lambda}} = \underset{\boldsymbol{\lambda}}{\operatorname{argmin}} \left\{ \sum_{t=1}^{n} \rho \left(\frac{y_t - \hat{y}_t^{t-1}(\boldsymbol{\lambda})}{s_t} \right) \right\}. \tag{64}$$

Although the estimates (63) and (64) were originally introduced for robust estimation of linear AR and ARMA models with additive outliers, their use for robust fitting of feedforward and recurrent network approximations to NAR and NARMA models is quite natural. Thus we focus on use of the simpler form (64) with the recurrent model predictor form

$$\begin{aligned} \hat{y}_t^{t-1}(\boldsymbol{\lambda}) &= g(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda}) \\ &= \sum_{i=1}^{I} W_i \sigma \left(\sum_{j=1}^{p} w_{ij} \hat{x}_{t-j} + \sum_{j=1}^{q} w'_{ij} \hat{\varepsilon}_{t-j} + \theta_i \right). \end{aligned} \tag{65}$$

We use $\rho = \rho_{\mathrm{H}}$ as given in Fig. 7(a). We describe computation of $\hat{s}_t(\boldsymbol{\lambda})$ shortly.

We now describe an iterative computational strategy for approximately solving the optimization problem (64), assuming for the time being that $\hat{\boldsymbol{x}}_t$ and s_t do not depend on $\boldsymbol{\lambda}$. Differentiating the right hand side of (64) with respect to $\boldsymbol{\lambda}$ give the estimating equation

$$\sum_{t=1}^{n} \boldsymbol{Dg}(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda}) \psi \left(\frac{y_t - g(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda})}{s_t} \right) = 0 \tag{66}$$

where the gradient $\boldsymbol{Dg}(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda})$ of $g(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda})$ is a column vector, and (66) is solved for the estimate $\hat{\boldsymbol{\lambda}}$.

Now consider the first element of the vector recursion (42) with $\hat{y}_t^{t-1} = f(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda})$ replaced by $\hat{g}(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda})$:

$$\hat{x}_t = g(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda}) + s_t \psi \left(\frac{y_t - g(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda})}{s_t} \right). \tag{67}$$

Multiplying both sides of (67) by $\boldsymbol{Dg}(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda})$ and summing gives

$$\begin{aligned} &\sum_{t=1}^{n} \boldsymbol{Dg}(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda}) s_t \psi \left(\frac{y_t - g(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda})}{s_t} \right) \\ &= \sum_{t=1}^{n} \boldsymbol{Dg}(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda}) [\hat{x}_t - g(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda})]. \end{aligned} \tag{68}$$

The left-hand side would be equivalent to (66) if s_t had a constant value. In fact s_t equals a robust estimate $\hat{\sigma}_\varepsilon$ of σ_ε for most t when most of the observations y_t are free of outliers (see [25]). Thus it is reasonable to assume that the left hand side of (68) is equal to (66), up to a constant multiplier.

On the right hand side of (68) is the estimating equation for the least-squares fitted network approximation based on the robustly filtered data. That is, the estimate $\hat{\boldsymbol{\lambda}}$ obtained by solving the right hand side of (68) is given by

$$\hat{\boldsymbol{\lambda}} = \underset{\boldsymbol{\lambda}}{\operatorname{argmin}} \sum_{t=1}^{n} (\hat{x}_t - g(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{\lambda}))^2. \tag{69}$$

In the above arguments leading to (69) we have been treating the $\hat{x}_t$ and s_t as fixed and independent of $\boldsymbol{\lambda}$. This is not quite ture, as the robust filter depends upon $\boldsymbol{\lambda}$. However, typically most values of $\hat{x}_t$ equal y_t and such values are essentially independent of $\boldsymbol{\lambda}$, as are the corresponding values of s_t. At gross outlier positions $\hat{x}_t = \hat{x}_t^{t-1}(\boldsymbol{\lambda})$ and $s_t = s_t(\boldsymbol{\lambda})$ depend upon $\boldsymbol{\lambda}$. Thus (69) should yield a good approximation to (66), at least when the fraction of outliers is not too large.

The above discussion motivates the following simple iterative strategy for computing the estimate $\hat{\boldsymbol{\lambda}}$:

1) Train a NARMA(p,q) recurrent network approximation via least squares to get an initial estimate $\hat{\boldsymbol{\lambda}}^0$ of the network parameters. Here we use: Backpropagation for NAR(p) models and a variant of the Williams and Zipser algorithm [41] for the NARMA(p,q) models.

   If the outliers correspond to the largest estimated residuals obtained from the estimated NARMA(p,q) model, the following algorithm produces a robust NARMA(p,q) model The model can be constrained to be smooth via Bayesian or Minimum Description Length techniques, the outliers are then less likely to be modeled explicitly. Alternatively, if a fairly small number of iterations M is used, outliers are not modeled. We have found that M in the range 100 to 1000 suffices for computing $\hat{\boldsymbol{\lambda}}$.
2) Compute a highly robust scale estimate $\hat{\sigma}_\varepsilon$ of σ_ε based on the initial prediction residuals $r_t^0 = y_t - \hat{y}_t^{t-1}(\hat{\boldsymbol{\lambda}})$ *with no* robust filtering. The standardized median absolute deviation about the median (MADM)

$$\hat{\sigma}_\varepsilon^0 = 1.483 \operatorname{MED} \{ |r_t^0 - \operatorname{MED} \{r_t^0\}| \}$$

   provides such an estimate. The standardization constant 1.483 insures that $\hat{\sigma}_\varepsilon$ is a consistent estimate of the true σ_ε when the ε_t are Gaussian. Note that as the percentage of outliers increases, reliability of the scale estimate goes down. But for up to 20% outliers, the MADM approach to robust scale estimates is reliable.
3) For $j = 0, 1, \cdots, J$
4) a. Filter the training data with a robust filter based on $\hat{\boldsymbol{\lambda}}^j$. This results in robustly filtered estimates $\hat{x}_t^j = \hat{x}_t(\hat{\boldsymbol{\lambda}}^j)$.
   b. Compute a new MADM robust scale estimate $\hat{\sigma}_\varepsilon^{j+1}$ based on the residuals $r_t^j = y_t - \hat{y}_t^{t-1}(\hat{\boldsymbol{\lambda}}^j)$

where the robust filter from a) has been used to construct the predictor $\hat{y}_t^{t-1}(\hat{\boldsymbol{\lambda}}^J)$

c. Use as training data the $\hat{x}_t^j, t = 1, \cdots, n$, to train a recurrent NARMA type predictor model via least squares for M iterations (here $M_{J+1} = 1000$ suffices). This results in new recurrent network parameter estimates $\hat{\boldsymbol{\lambda}}^{J+1}$. Go to (a).

5) We have found that typically $J = 25$ to 100 iterations of the above outer loop suffices.

1) Tuning Constants: With the above algorithm we do not make explicit use of the loss function ρ in (66), so we do not need to specify any tuning constant for ρ. The ψ function we use for the robust filter (40)–(44) (with $\boldsymbol{f}$ replaced by $\boldsymbol{g}$) is the Hampel two-part redescending function of Fig. 6. This ψ function has two tuning constants, a and b, which provide a trade-off between statistical efficiency of estimation when no outliers are present and robustness when outliers are present. Our experience indicates that the values $a = 2$ and $b = 3$ work well for a wide range of applications.

C. Comment on Estimation Maximization (EM) Algorithm

The iterative approach of filtering for outliers and re-estimating parameters is a special case of the EM algorithm first presented in [9]. The EM algorithm has been applied recently to hierarchical mixture of experts models, [18] and [31] among others. This is referred to as a *soft* filtering approach because a data point can effect more than one model. A *hard* filtering approach would classify a data point as either in one class or the other, MARS and CART are examples of hard filtering.

The model we have described has two data classes, normal and outlier data. Our approach classifies points with small residuals as being normal and those ponts with extremely large residuals as being outliers. There is a gray area where the data is not classified as normal or outlier. In this case, the data point influences the model but the influence is down weighted. Because of the gray area, our algorithm is an example of *soft* filtering.

It should also be noted that the algorithm can be done on-line if the parameters and the scale are both calculated recursively, see [18] for a related discussion in the context of hierarchical mixture models.

D. NAR(1) Robust Fitting Example

In this section we provide an example of applying the robust neural network procedure of Sections VI-A and VI-B to time series with additive outliers, namely the 10% additive outlier example of equations (24), (25), and (30). See also Fig. 5(a) and (b).

Fig. 8 displays the lag-one scatterplot for the series of Fig. 5(a), along with:

1) The true function g (solid line) given by (27)
2) The least squares estimated NAR(1) predictor $\hat{g}$ based on the y_t as given by (24), (25), and (30) with $\gamma = 0.1$ corresponding to 10% outliers.

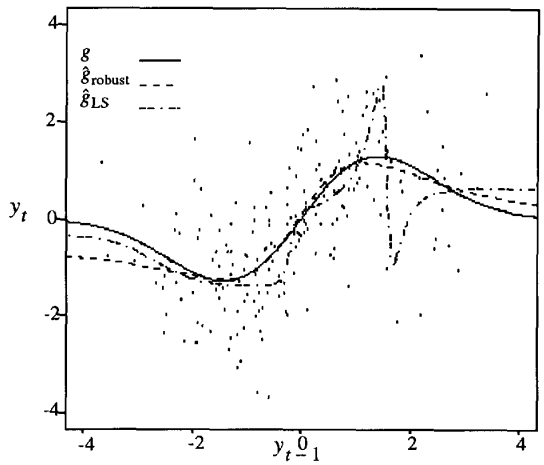


Fig. 8. Lag one, scatter plot for 10% outlier data with the true, and least squares and robust NAR(1) estimates.

3) The robustly estimated NAR(1) predictor $\hat{g}_{\text{robust}}$ based on the y_t as given by (24), (25), and (30) with $\gamma = 0.1$ corresponding to 10% outliers.

VII. Competition on Load Forecasting Data

Neural networks have been applied to electric load forecasting with some success. See for example, [8], [22], [32], and [42]. A recurrent network modeled the data that came from a competition to predict the loads of the Puget Sound Power and Light Company from November 11, 1990 to March 31, 1991. The object was to predict the hourly demand for the electric power. The predictions were made in the following manner, which was dictated by economic considerations. At 8:00 A.M. on each working day, Monday through Friday, a prediction is made for each hour of the following working day, commencing with 1:00 A.M. and ending at 12:00 P.M. With regard to predictions made on Friday, the next working day is Monday.

The predictor variables available for use in our predictor model are:

1) Power demand (or load) l and temperature T up to and including 8:00 A.M., the time at which predictions are made for the next day
2) The weatherman's prediction made at 8:00 A.M. each day for every hour of the next day to be predicted. Monday forecasts are made on Friday, Tuesday through Friday forecasts are made on the preceding day.

The forecasting competition was based on prediction performance during the period Nov. 11, 1990 to March 31, 1991. The historical time series data for load, temperature and temperature predictions by the weatherman for the same intervals (November 11 to March 31) during each of the four years 1986–7 through 1989–1990 were available. In Figs. 9 and 10 we display this time series data. Fig. 9 displays all of the 9:00 A.M. data for 1987, 1989, and 1991, superimposed in a single plot. This allows one to gauge the overall trend effect across the years 1987–1991. Fig. 10 displays the power

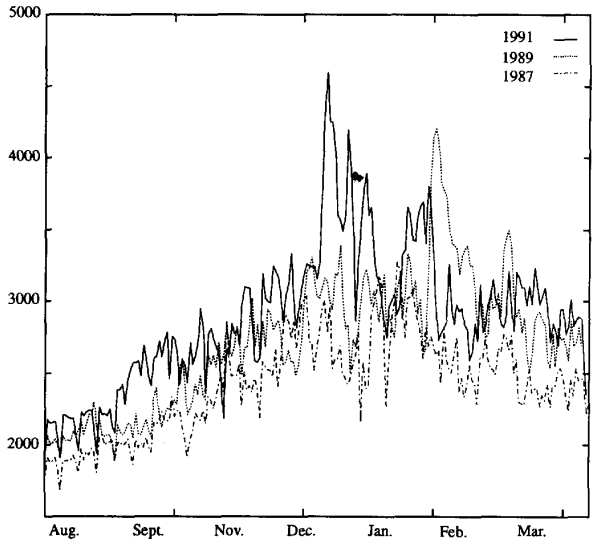


Fig. 9. 9:00 A.M. power demand for 1991, 1989, and 1987.

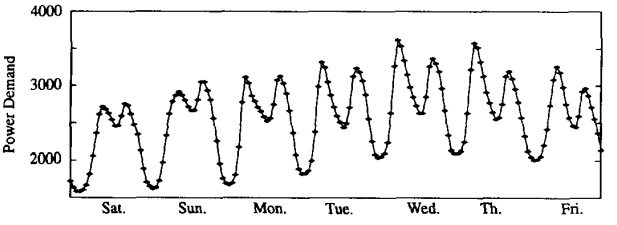


Fig. 10. Hourly power demand for the week of 1/20/91.

demand hour by hour over a one week interval. This plot shows quite clearly the diurnal and weekly non-stationary cycle behavior of the data.

Because of the strong diurnal cycle we decided to build a separate model for each hour to be predicted. In order to model the weekly cyclic and seasonal trend/cycle fragment, we include a day of the week variable $dw = 1, 2, 3, 4, 5$ and day of the year variable $dy = 1, 2, \cdots, 222$ (to cover Nov. 11 to March 31 of each year). To model any trend from year to year, we included a year variable $k = 1, 2, 3, 4, 5$ corresponding to 1986–7, $\cdots$, 1990–1.

The prediction performance results are presented for the following "NARMAX" models (NARMA(1, 1) plus exogeneous temperature and temperature prediction variables):

For hours $t - 1, 2, \cdots, 8$

$$l_{k,t} = f_t(l_{k-1,t}, T_{k-1,8}, \hat{T}_{k,t}, dw, k, y, e_{k-1,t}) + e_{k,t}$$

Tuesday–Friday

$$l_{k,t} = f_t(l_{k-3,t}, T_{k-1,8}, \hat{T}_{k,t}, dw, k, y, e_{k-3,t}) + e_{k,t}$$

Monday

For hours $t = 9, 10, \cdots, 24$

$$l_{k,t} = f_t(l_{k-2,t}, T_{k-1,8}, \hat{T}_{k,t}, dw, k, y, e_{k-2,t}) + e_{k,t}$$

Wed.–Friday

$$l_{k,t} = f_t(l_{k-4,t}, T_{k-1,8}, \hat{T}_{k,t}, dw, k, y, e_{k-4,t}) + e_{k,t}$$

Tuesday

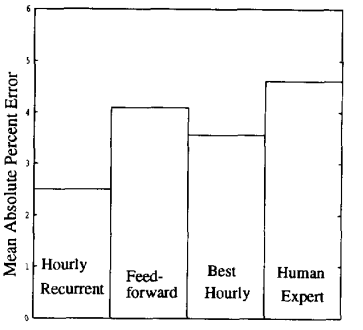


Fig. 11. 1991 A.M. peak mean square error.

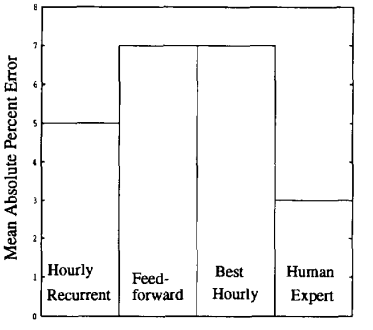


Fig. 12. 1991 peak mean square error.

$$l_{k,t} = f_t(l_{k-3,t}, T_{k-1,8}, \hat{T}_{k,t}, dw, k, y, e_{k-3,t}) + e_{k,t}$$

Monday

Weekend results were not considered economically relevant. The weekday predictions are evaluated on four criteria which can be described in terms of the Mean Absolute Percentage Error (MAPE) function over the hours h_1 to h_2,

$$\mathrm{MAPE}(h_1, h_2) = \frac{1}{N} \sum_{k=1}^{N} \frac{\sum_{i=h_1}^{h_2} |l_{k,i} - \hat{l}_{k,i}|}{\sum_{i=h_1}^{h_2} |l_{k,i}|} \times 100\%.$$

The most important are the Mean Absolute Percentage Error (MAPE) of the A.M. peak, 7 A.M. to 9 A.M., $E_{\mathrm{A.M.}} = \mathrm{MAPE}(7, 9)$, and the MAPE of the P.M. peak, $E_{\mathrm{P.M.}} = \mathrm{MAPE}(17, 19)$. Two other criteria, the total error, $E_{\mathrm{Total}} = \mathrm{MAPE}(1, 24)$, and the winter peak error

$$E_{\mathrm{Peak}} = \frac{|l_{k,i} - \hat{l}_{k,i}|}{|l_{k,i}|} \times 100\%$$

where $l_{k,i} = \max l$ are also considered valuable. $E_{\max}$ tends to be the most difficult measure, because the peak tends to be outside the training set. The performance of the recurrent network on two of these error measures is shown in Figs. 11 and 12, for other error measures see [7].

The predictions for the 9 A.M. time series is shown in Fig. 13. The effect of the nonlinear nature of neural networks was apparent in the error residuals of the test set. Figs. 14 and 15 are plots of the residuals against the predicted load for

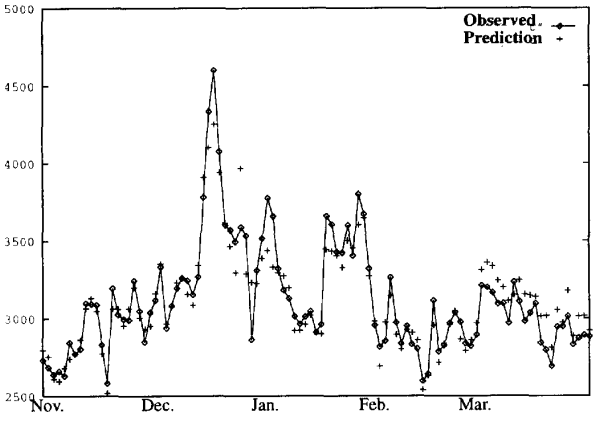


Fig. 13. Actual and predictions, 1991 test set.

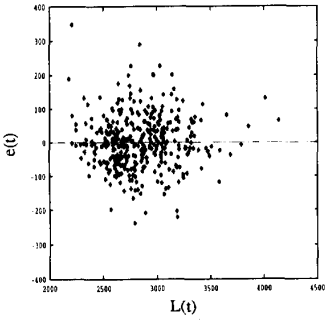


Fig. 14. Residual vs. predictions, 1986–1990.

the training and testing set of the hourly recurrent network. Fig. 14 shows the desired behavior, residuals and predictions that are uncorrelated for the training set. However in Fig. 15, the residuals tend to be positive for larger loads and negative for lesser loads for the test set. Fig. 16 and 17 are plots of residuals versus the previous residuals for the training and testing set. The residuals are uncorrelated for the training set, Fig. 16, but the testing set, Fig. 17, shows definite skewing behavior. The skewing behavior in Fig. 15 and 17 is a product of the squashing effect of the sigmoidal nonlinearities. This effect could be removed by adding a linear component to the predictor. The squashing effect becomes acute during the prediction of the peak loads of the winter. These peak loads are caused when a cold spell occurs and the power demand reaches record levels. As shown in Fig. 12, this is the only measure on which the performance of the recurrent networks is surpassed.

VIII. Robust One Step Puget Power Predictions

In Section VII, the Puger Power Demand time series was modeled for purposes of forecasting power demand. The robust algorithms developed in Section VI are now used on the 8 A.M. Power Demand time series data. This time of day corresponds to nearly peak power demand for each day. An

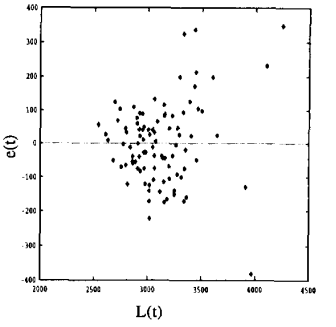


Fig. 15. Residuals vs. predictions, 1991.

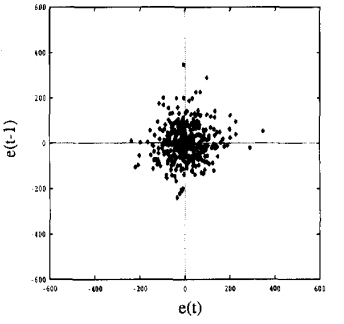


Fig. 16. Lag 1 scatter plot, 1986–1990.

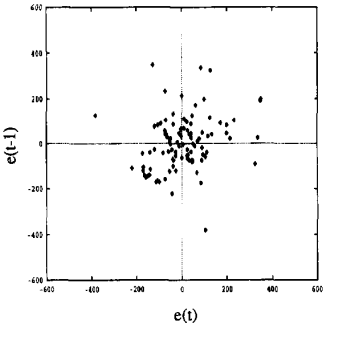


Fig. 17. Lag 1 scatter plot, 1991.

NARX(1) model was fitted to the data

$$l_t = f(l_{t-1}, T_{t-1}, \hat{T}_t, dw, y) + e_t + v_t$$
$$y_t = l_t + v_t \tag{70}$$

where l_t represents the electric power demand on day t, T_t is the temperature at day $t, \hat{T}_t$ is the forecasted temperature for day $t. dw$ is the day of the week, y is the year, e_t is the random noise at 8:00 A.M. on day t. The observation (70) equation contains v_t: which models the existence of outliers. Note that the variables $T_t, \hat{T}_t, dw$ and y are so-called *exogenous* variables. Hence the term NARX(1) rather than NAR(1).

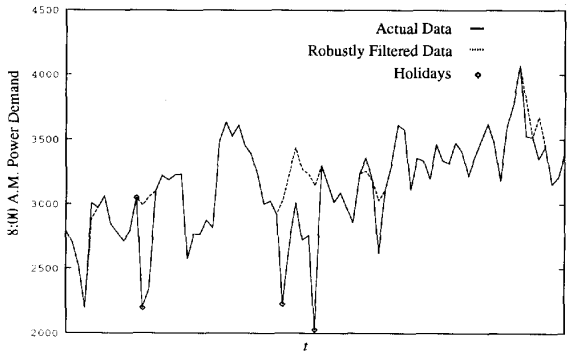


Fig. 18. Observed and robustly filtered 1990 filtered training data.

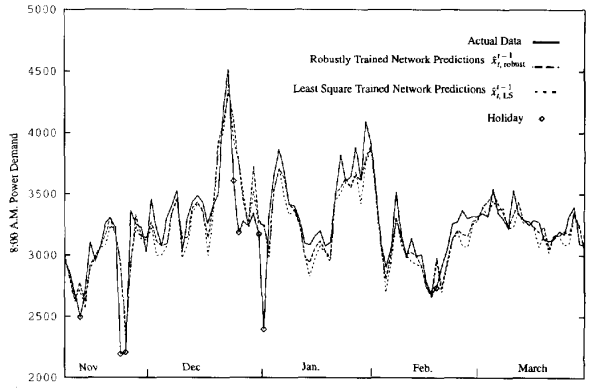


Fig. 19. Puget power demand data for 1991 competition data and predictions based on LS and Robust Network fits.

Fig. 18 shows the original 8:00 A.M. power demand time series for the year 1990, which was used to train a 5 hidden unit feedforward network with the robust method of Section VI, along with the robustly filtered data based on the robustly trained network.

Note that the robust filter values $\hat{x}_t$ agree with the observed data y_t most of the time (recall that we claimed in Section IV that the typical robust filter behavior has this general character). Some (but not all) of the times at which $\hat{x}_t$ and y_t differ are holidays. Typically a holiday causes a large magnitude because holidays result in exceptionally low power demand. Notice also that when the robust filter "interpolates" (i.e., provides pure predictions for $\hat{x}_t$) in the vicinity of outlier positions, it produces qualitative behavior similar to the rest of the unaltered series where $\hat{x}_t = y_t$. This is of course natural since the robustly trained neural network models well the dynamics of the outlier free positions of the time series.

Fig. 19 shows the 1991 Puget Power Demand time series, along with one-day ahead predictions/forecasts ($\hat{x}^{t-1}_{t,\mathrm{LS}}$ and $\hat{x}^{t-1}_{t,\mathrm{robust}}$) based on the classic least squares trained network and our robustly trained network, respectively. It is clear from this figure that the predictions $\hat{x}^{t-1}_{t,\mathrm{robust}}$ based on the robustly trained neural network tend to be closer to the true values than the predictions $\hat{x}^{t-1}_{t,\mathrm{LS}}$ based on the classic least squares network fit.

The differences in performance between $\hat{x}^{t-1}_{t,\mathrm{robust}}$ and $\hat{x}^{t-1}_{t,\mathrm{LS}}$ are displayed in Table IV using two measures of variability of the prediction errors

$$r_t = y_t - \hat{x}^{t-1}_t \tag{72}$$

namely:

1) Mean-squared error (MSE). This is the usual measure of prediction error performance, $\mathrm{MSE} = (1/n)\sum_{t=1}^{n} r_t^2$. However, its value is inflated by the presence of outliers.
2) Median of square error (Median of SE). This measure of prediction error is based on the sample median of the squared residuals, Median $\{r_1^2, \cdots, r_n^2\}$. Hence this measure is not inflated by outliers, and gives a more accurate indication of prediction performance for the bulk of the data that is free of outliers.

TABLE IV
PERFORMANCE OF PREDICTORS $\hat{x}^{t-1}_{t,\mathrm{LS}}$ AND $\hat{x}^{t-1}_{t,\mathrm{robust}}$ FOR TRAINING AND TEST DATA.

Training Method	1990 Training Data		1991 Test Data	
	M.S.E.	Median S.E.	M.S.E.	Median S.E.
Least Squares	.0248	.0140	.0230	.0187
Robust	.0351	.0086	.0167	.0114

Notice that on the 1991 training data the MSE measurement indicates that $\hat{x}^{t-1}_{t,\mathrm{LS}}$ is considerably better than the $\hat{x}^{t-1}_{t,\mathrm{robust}}$, whereas its opposite is true using the Median of SE. The fact that for both $\hat{x}^{t-1}_{t,\mathrm{LS}}$ and $\hat{x}^{t-1}_{t,\mathrm{robust}}$ the Median of SE is smaller than the MSE for both 1990 training and 1991 test data reflects the fact that the MSE is inflated by the outliers in all the cases. On the 1991 test data set $\hat{x}^{t-1}_{t,\mathrm{robust}}$ is better than $\hat{x}^{t-1}_{t,\mathrm{LS}}$ by about the same relative amount using both MSE and Median of SE.

Overall the robust neural network fitting method yields considerably improbed prediction performance relative the classical LS network fitting.

A few passing comments on outliers due to holiday effects in closing: It is well known in the statistical time series literature that the holiday effects can be handled by a structured dummy variable approach (called "intervention analysis" by Box and Tiao, 1975). This method could be adopted to time series neural network fitting. Our experience indicates that the robust network fitting method we propose and the structured dummy variable method yields similar results in the network fitting stage when holiday effects cause outliers. However, the dummy variable approach has the advantage of providing possible considerable improved prediction for future holidays. We hope to explore this issue further is a subsequent paper.

IX. CONLCUSION

Recurrent networks were shown to be a special case of nonlinear autoregressive moving average models, abbreviated NARMA. Feedforward networks were shown to be a special

case of nonlinear autoregressive models, abbreviated NAR. This being the case, recurrent networks are well-suited for time series that possess moving average components. Good performance for these predictor models were demonstrated on synthetic time series.

Neural networks fit by classical least squares were shown to be highly sensitive to outliers. In the case of a small number of outliers, the effect on the predictor tends to be localized. Our studies show that for single outliers, neural networks tend to model gross outliers completely. In such cases, the model is a useful predictor as long as the predictions are not near the outliers. However, very bad predictions can occur in localized regions of the parameter space.

In cases where there is more than a small number of outliers (e.g., 5 to 20% of outliers), the classical least squares fitting of neural network models gives serious overall distortion to the neural network model. We introduced a robust filtering algorithm as an integral part of the training. The filtering gives rise to very good distortion free estimation/fitting of the neural network model. The essence of the robust filter is to provide (one sided) interpolated value at gross outlier positions.

It was shown that recurrent neural networks can give superior results for load forecasting, but as with other models, the input configuration is critical to good prediction performance. The relative superiority of recurrent networks to feedforward networks in load forecasting is not just due to its ability to model time series data with lower errors, but rather to model a training set parsimoniously. Robust filtering based training were shown to yield considerable improvement over conventional least squares fitting of neural network models for electric load forecasting.

REFERENCES

[1] B. Anderson, and J. Moore, *Optimal Filtering*. Englewood Cliffs, NJ: Prentice-Hall, 1979.

[2] R. Battiti, "First- and second-order methods for learning: Between steepest descent and Newton's method," *Neural Computation*, vol. 4, pp. 141–166, 1992.

[3] G. E. Box and G. M. Jenkins, *Time series analysis: forecasting and control*, Holden-Day, 1976.

[4] G. E. Box and G. C. Tiao, "Intervention analysis with applications to economic and environmental problems," *J. Amer. Stat. Assoc.*, vol. 70, 1975.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group, 1984.

[6] D. R. Brillinger, "The identification of polynomial systems by means of higher order spectra," *J. Sound Vib.*, vol. 12, pp. 301–313, 1970.

[7] J. T. Connor, "Time series and neural network modeling," doctoral dissertation, Univ. of Washington, 1993.

[8] J. T. Connor, L. E. Atlas, and R. D. Martin, "Recurrent networks and NARMA modeling," *Advances in Neural Information Processing Systems 4*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds. San Matteo, CA: Morgan Kauffman, pp. 301–308, 1992.

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, B*, vol. 39, pp. 1–38, 1977.

[10] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–21, 1990.

[11] J. H. Friedman, "Multivariate adaptive regression splines," *The Annals of Statistics*, vol. 19, pp. 1–141, 1991.

[12] J. H. Friedman and W. Stuetzle, "Projection Pursuit regression," *J. Amer. Statist. Assoc.*, pp. 817–823, 1981.

[13] S. Grossberg, "Adaptive pattern classification and universal recoding: 1. Parallel development and coding of neural feature detectors," *Biol. Cybern.*, vol. 23, pp. 121–134, 1976.

[14] A. C. Harvey, *The econometric analysis of time series*. Cambridge, MA: MIT Press, 1990.

[15] G. E. Hinton, and T. J. Sejnowski, "Learning and relearning in Boltzman machines," D. E. Rumelhart, J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, Cambridge, MA: MIT Press, Chapter 7, 1986.

[16] J. J. Hopfield "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554–2558, 1982.

[17] A. Jazwinski, *Stochastic processes and filtering theory*. New York: Academic Press, 1970.

[18] M. I. Jordan, "Attractor Dynamics and parallelism in a connectionist sequential machine," in *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ: Erlbaum, 1987, pp. 531–546.

[19] B. Kleiner, R. D. Martin, and D. J. Thomson, "Robust estimation of power spectra," *J. Roy. Stat. Soc. Ser. B* vol 41, pp. 313–351, 1979.

[20] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Englewood Cliffs, NJ: Prentice Hall, 1992.

[21] A. Lapedes, and R. Farber, "Nonlinear signal processing using neural networks: Prediction and Modeling," Technical Report, LA-UR87-2662, Los Alamos National Laboratory, Los Alamos, New Mexico, 1987.

[22] D. J. C. MacKay, "Bayesian non-linear modeling for the energy prediction completion," Draft 1.3., 1993.

[23] R. D. Martin, and R. Fraiman, "Approximate conditional mean non-Gaussian filters," *Tech. Report*, Dept. of Statistics, Univ. of Washington, 1993.

[24] R. D. Martin, "Robust estimation of autoregressive models (with discussion)," in *Directions in Time Series*, D. R. Brillinger and G. C. Tiao, Eds. Institute of Mathematical Statistics, Hayward, Calif., 1980, pp. 228–262.

[25] R. D. Martin, and D. J. Thompson, "Robust-resistant spectrum estimation," *Proc. IEEE*, vol. 70, pp. 1097–1115, 1982.

[26] R. D. Martin and V. J. Yohai, "Highly robust estimation of autoregression integrated time series models," *Tech. Report.*, 1985.

[27] R. D. Martin, "Robust estimation for time series autoregressions," in *Robustness in Statistics*, R. L. Launer and G. N. Wilkinson, Eds. New York: Academic Press, pp. 147–176, 1979.

[28] C. J. Masreliez, "Approximate non-Gaussian filtering with linear state and observation relations," *IEEE Transactions on Automatic Control*, pp. 107–110, 1975.

[29] J. S. Meditch, *Stochastic Optimal Linear Estimation and Control*. New York: McGraw-Hill, 1969.

[30] M. C. Mozer, "Neural net architectures for temporal sequence processing," in *Predicting the Future and Understanding the Past*, A. Weigend and N. Gershenfeld, Eds. Redwood City, CA: Addison-Wesley.

[31] S. J. Nowlan, "Soft competitive adaptation: Neural network learning algorithm based on fitting statistical mixtures," Tech. Report, CMU-CS-91-126, CMU, Pittsburgh.

[32] D. C. Park, M. A. El-Sharkawi, and R. J. Marks, "An adaptively trained neural network," *IEEE Trans. on Neural Networks*, vol. 2, no. 3, 1991.

[33] F. J. Pineda, "Generalization of backpropagation to recurrent and higher order networks," in *Proc. IEEE Conf. Neural Inform. Proc. Syst.*, 1987.

[34] A. J. Robinson and F. Fallside, "The utility driven dynamic error propagation network," Tech Report CUED/F-INFENG/TR.1, Cambridge: Cambridge University, Department of Engineering.

[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, vol. 1, D. E. Rumelhart, and J. L. McCelland, Eds. Cambridge, MA: MIT Press, pp. 318–362, 1986.

[36] J. Schmidhuber, "A fixed size storage $O(n^3)$ time complexity learning algorithm for fully recurrent continually running networks," *Neural Computation*, vol. 4, pp. 243–248, 1992.

[37] T. Subba Rao, "On the theory of bilinear models," *J. Roy. Statist. Soc. Ser. B*, vol. 43, pp. 244–255, 1981.

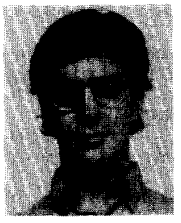
[38] H. Tong, *Threshold Models in Non-linear Time Series Analysis*. New York: Springer-Verlag, 1983.

[39] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time delay neural networks," *IEEE Trans. on Acous. Speech and Sig. Proc.*, vol. 37, pp. 324–339, 1989.

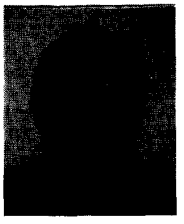
[40] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. of the IEEE*, pp. 1550–1560, vol. 78, no. 10, Oct. 1990.

[41] R. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270–280, 1989.

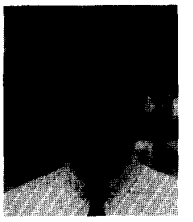
[42] Jen-Lun Yuan and T. L. Fine, "Forecasting demand for electric power," *Advances in Neural Information Processing Systems* vol. 5, S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds. San Matteo, CA: Morgan Kauffman, pp. 739–746, 1993.



Jerome T. Connor received his B.S.E.E. degree from the University of Delaware, his M.S. degree in Physics from the Universtiy of California at Los Angeles and a Ph.D. degree from the University of Washington. His Ph.D. dissertation was on neural networks and time series prediction. He is currently a postdoc at Bellcore. He is currently investigating robust nonlinear modeling for predicting time series in the telecommunications industry.



R. Douglas Martin (S'57-M'65-S'67-M'69), received his B.S.E. and Ph.D. in Electrical Engineering from Princeton University in 1959 and 1969, respectively. He was a faculty member in the Department of Electrical Engineering at the University of Washington, Seattle, WA from 1969 to 1980. In 1980 Martin joined the newly formed Department of Statistics at the University of Washington. From 1973 to 1984 he was a consultant in the Mathematics and Statistics Research Center at Bell Labs. From 1987 to 1993 he was the CEO (and founder) of Statistical Sciences, Inc. (StatSci), a company that develops and markets software products and services for graphical data analysis, statistical modeling, and mathematical computing (StatSci's flagship product is S-PLUS). Martin is currently on leave from the University of Washington and serving as the Chief Technical Officer and Vice President of Research and Development for the StatSci Division of Math-Soft. Professor Martin's research interests include robust methods, time series, statistical computing, wavelets, and neural networks. He is an author or co-author of many papers in these areas, including two Royal Statistical Society Discussion Papers ("Robust Spectral Analysis" and "Diagnostics for Time Series") and an invited Annals of Statistics Discussion paper on "Influence Functionals for Time Series."



Les. E. Atlas (S'78-M'82), received his B.S.E.E. degree from the University of Wisconsin and his M.S. and Ph.D. degrees from Stanford University in 1979 and 1984, respectively. His Ph.D. dissertation was on the design of speech processors for an auditory prosthesis for the profoundly deaf. He joined the University of Washington Department of Electrical Engineering in 1984 and is currently an Associate Professor of Electrical Engineering. He co-founded the Interactive Systems Design Laboratory at the University of Washington and is currently doing research in acoustic signal processing and recognition, neural network classifiers, and biologically-inspired signal processing algorithms and architectures. Prof. Atlas' research is funded by the Washington Technology Center and Boeing Commercial Airplane Company. Dr. Atlas was a 1985 recipient of a National Science Foundation Presidential Young Investigator Award and he has served as a consultant for many industrial projects including those with Honeywell Marine Systems, the Boeing Company and Advanced Technology Laboratories. He also was General Chair of the 1992 IEEE Symposium on Time/Scale Analysis.