

Sequential Network Construction for Time Series Prediction

Tomasz J. Cholewo Jacek M. Zurada

University of Louisville

Department of Electrical Engineering

Louisville, KY 40292

email: t.cholewo@ieee.org, j.zurada@ieee.org

tel: (502) 852-6314, fax: (502) 852-6807

Abstract

This paper introduces an application of the Sequential Network Construction (SNC) method to select the size of several popular neural network predictor architectures for various benchmark training sets. The specific architectures considered are a FIR network and the partially recurrent Elman network and its extension, with context units also added for the output layer. We consider an enhancement of a FIR network in which only those weights having relevant time delays are utilized. Bias-variance trade-off in relation to the prediction risk estimation by means of Nonlinear Cross Validation (NCV) is discussed. The presented approach is applied to the Wölfer sunspot number data and a Mackey-Glass chaotic time series. Results show that the best predictions for the Wölfer data are computed using a FIR neural network while for Mackey-Glass data an Elman network yields superior results.

1. Introduction

A number of different NN architectures and learning methods have been applied to the time series prediction problem with varying degrees of success. Both Elman and FIR networks were previously used for forecasting [1, 2] with the latter having a distinction of being the winning entry in a Santa Fe competition [3]. However, there still exists a need for satisfactory guidelines when choosing a model size for a particular prediction problem. In this paper we propose an application of the Sequential Network Construction (SNC) [4] method to neural network predictor architecture selection.

The objective of NN predictor training is to minimize the cost function $E = \sum_{\tau=1}^T \sum_{k=1}^{N^L} e_k^2(\tau)$ where $e_k(\tau) = d_k(\tau) - o_k(\tau)$ is the difference between desired and calculated values at output k at time step τ . The sums are taken over all T time instants in the training sequence and over all N^L outputs, respectively. The

simplest method of adapting weights towards the minimum of E is the steepest descent method in which weights are changed in the direction of the negative error gradient scaled with a “learning rate” η according to the equation $\Delta w_{ij} = -\eta \nabla_{w_{ij}} E$. Weight adaptation in this paper is performed with a scaled conjugate gradient method [5] which significantly improves speed and convergence of training as compared to standard techniques. Weights are modified at the end of each training epoch (i.e., in batch mode). This is opposed to stochastic adaptation which performs weight updates after each pattern presentation and thus represents only an approximation to the true gradient. This consideration is especially important in case of networks with an internal memory where algorithms differing in weight update order can exhibit significantly different performance [6].

2. Time Series Modeling Architectures

2.1. FIR neural network

The Finite Impulse Response (FIR) network as proposed in [2] is a modification of the basic multilayer feedforward network in which each weight is replaced by a FIR linear filter. Output of a single filter

$$y(k) = \sum_{n=0}^T w(n)x(k-n)$$

corresponds to the moving average component of the ARMA model [7]. This modification is implemented by substituting each scalar weight w_{ij}^l connecting the output of neuron j in layer l to the input of neuron i in layer $l+1$ by a vector $\mathbf{w}_{ij}^l = [w_{ij,0}^l, w_{ij,1}^l, \dots, w_{ij,T^l}^l]$ where T^l is the number of time lags in layer l . Output of each neuron can then be described by $x_i^{l+1}(\tau) =$

$f_i(s_i^{l+1}(\tau))$ where

$$s_i^{l+1}(\tau) = \sum_{i=1}^{N_l} \sum_{k=0}^{T^l} w_{ijk}^l(n) x_j(\tau - k),$$

N^l is a number of neurons in layer l , and f_i is an activation function of neuron i .

A NN structure commonly used for prediction is a multilayer perceptron with time lagged inputs. Its generalization produces a Time Delay Neural Network (TDNN) in which both inputs and hidden layer outputs are buffered several time steps and then input to following layers. The FIR network is functionally equivalent to a TDNN but uses a different weight update method.

Calculation of the error gradient in a FIR network is performed by means of “temporal backpropagation” [2]:

$$\nabla_{w_{ijk}^l} E = \sum_{\tau=k}^T \delta_j^{l+1}(\tau) x_{ik}^l(\tau - k)$$

where the local gradient for neuron j in layer l is

$$\delta_j^l(\tau) = \begin{cases} -2e_j(\tau) f_j'(s_j^l(\tau)), & l = L, \\ f_j'(s_j^l(\tau)) \gamma_j^l(\tau), & 1 \leq l \leq L - 1 \end{cases}$$

and

$$\gamma_j^l(\tau) = \sum_{m=1}^{N^{l+1}} \sum_{k=0}^{T^{l-1}} w_{jmk}^l \delta_m^{l+1}(\tau + k).$$

The original FIR network has weights for each time lag from 0 to T^l in layer l . This leads to a large number of weights when large time lags are used to allow for the dependence of the current output on events from the past. This is required in the case of a time series with long term periodicity. Networks with too many weights usually display poor generalization capabilities unless additional complexity control techniques like early stopping [2] or pruning are used.

We propose a modification of a FIR NN architecture in which there are weights at desired time lags only. Denote such networks by $\text{FIR}(L_1, L_2)$ where L_i is a set of time lags employed in layer i . In this manner only important time lags are left in the architecture while unnecessary time lags are omitted. The detection of important time lags is left to heuristics.

2.2. Partially Recurrent Neural Networks

Partially recurrent networks [8] (also known as Elman-Jordan networks) are a subclass of recurrent networks. These are multilayer perceptron networks augmented with one or more additional context layers which store output values of one of the layers delayed

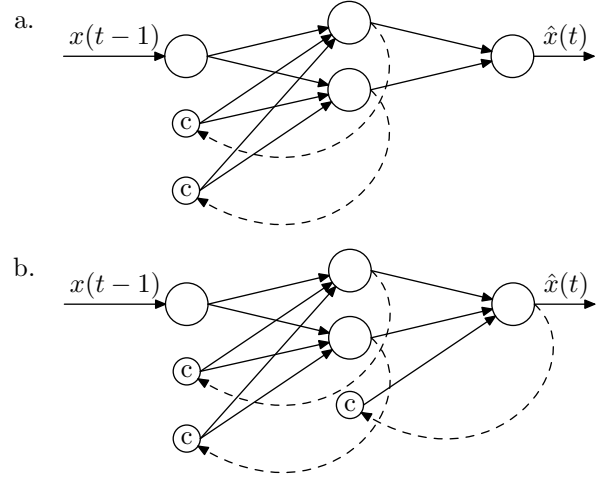


Figure 1. Examples of partially recurrent NNs with two hidden neurons: a. Elman; b. extended Elman (dashed lines represent unit delays, small circles are linear context units).

by one step. These layers are used for activating this or some other layer in the next time step. Two of several types of these networks which differ in the type of feedback used are shown in Figure 1. The Elman network has context units which store delayed hidden layer values and present these as additional inputs to the network, and an extended Elman network is an Elman network with an additional output-to-output feedback.

Several algorithms for calculation of the error gradient in general recurrent networks exist. Backpropagation Through Time (BPTT) [9] is an exact and fast training method. Its only relative disadvantages are substantial memory consumption since a history of network activations for all inputs has to be stored and that it is not suitable for on-line training. According to BPTT, the error gradient is calculated as follows:

$$\nabla_{w_{ik}} E = \sum_{\tau=1}^T \delta_i(\tau) x_k(\tau - 1)$$

where the local gradient for neuron j

$$\delta_j(\tau) = \begin{cases} -2e_j(\tau) f_j'(s_j(\tau)), & \tau = 0, \\ f_j'(s_j(\tau)) \gamma_j^l(\tau), & 0 < \tau < T \end{cases}$$

and

$$\gamma_j^l(\tau) = -2e_j(\tau) + \sum_{l=1}^{N^l} w_{lj} \delta_l(\tau + 1).$$

Since architectures of Elman networks are specific, simplified realizations of general recurrent networks, a

modified version of BPTT algorithm is used in which a pattern propagates from the input to the output layer in one time step.

3. Network Size Selection

No general guidelines are available in the technical literature as how to select the sizes of NN predictors hidden layers. This limitation leads to time consuming trial-and-error procedures. To circumvent this disadvantage we use a more systematic manner of determining appropriate architectures. A sequential network construction (SNC) algorithm [4] is employed to select the number of hidden neurons for each of the NN models considered. First, a network with a small number of hidden neurons is trained. Then a new neuron is added with weights randomly initialized. The network is retrained with changes limited to these new weights. Next all weights in the network are retrained. This procedure, repeated until the number of hidden neurons reaches a preset limit, substantially reduces training time in comparison with time needed for training of new networks from scratch. More importantly, it creates a nested set of networks having a monotonously decreasing training error and provides some continuity in the model space which makes a prediction risk minimum more easily noticeable.

The quality of a predictor can be defined as its ability to predict novel observations. A simple way to estimate prediction error when sufficient data is available is to divide it into two sets and use only one of them for training. The prediction error on the withheld (test) set is then an estimate of the prediction risk. When data is scarce techniques of sample re-use can be applied. The best known method here is cross-validation, requiring splitting the data into several disjoint subsets of roughly equal size, disregarding one subset and finding the predictor which minimizes the error for this training set. This procedure is repeated by putting aside and finding the test error for each subset in turn. The average value of these errors is the cross-validation average squared error. The existence of several local minima in the training error surface for nonlinear models causes predictors found in the process of cross-validation to correspond to different minima. To estimate the prediction risk associated with a minimum in which the predictor will actually perform, a modification of the standard CV procedure called Non-linear Cross Validation (NCV) [4] can be applied. It uses weights of a network trained on all available data rather than random initial weights as a starting point for retraining on successive subsets.

4. Examples and Discussion

The presented methodology has been applied to two time series: the Wölfer sunspot numbers and the Mackey-Glass chaotic time series. The task is to predict future values in an iterative (multistep) mode, that is, using the values predicted in the previous time steps as inputs for subsequent predictions.

Data is first normalized by a linear transformation $x_\tau^{\text{norm}} = (x_\tau - \bar{x})/\sigma_x$ to zero mean and unit variance. Weights are initialized on a neuron-by-neuron basis by setting them to random numbers from a uniform distribution with range $(-1/F_i, 1/F_i)$, where F_i is the fan-in (the total number of inputs) of neuron i in the network. Prediction quality is measured by means of the average relative variance $arv = \sum_{\tau=1}^T (x_\tau - \hat{x}_\tau)^2 / \sum_{\tau=1}^T (x_\tau - \bar{x})^2$ which is also known as normalized mean squared error (NMSE). A value of $arv = 0$ indicates perfect prediction while a value of 1 corresponds to simply predicting the average. We calculate arv on a separate withheld test set to avoid artificially optimistic estimates resulting from the fact that the chosen architecture is optimized specifically to perform well on the validation set.

All simulations of the studied time series were performed ten times, each time beginning with different initial conditions. Model selection results were averaged to reduce the influence of model variability on network size selection by introducing the possibility of escaping local minima.

4.1. Wölfer sunspot data

The annual Wölfer sunspot number time series is probably the most often cited time series and has therefore been selected for benchmarking in this paper. The values for the years 1770–1869 are used as the training/cross-validation set and the interval 1870–1889 as the test set.

In the first step a linear ARMA model is fit to the data. Box and Jenkins [7] identified the optimal model for this data as AR(2); however, they also stated that this is not a particularly good fit. Analysis of the residuals shows a peak at time lag 11 which corresponds to the main periodicity of the sunspot series. An AR model with three time lags 1, 2 and 11 gives better results. A prediction for 20 years ahead gives $arv = 0.445$ and 0.252 for the above two models respectively.

The best FIR network is found by using the SNC methodology for over a dozen tap configurations chosen heuristically on the basis of their performance in the ARMA model and the strong period 11 periodicity observable in the time series. Architectures considered are limited to single hidden layer networks because of their proven universal approximation capabilities and

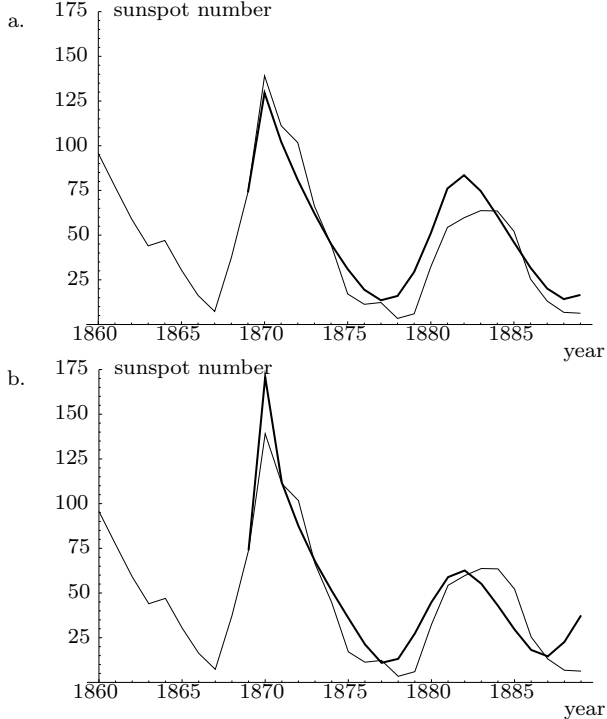


Figure 2. Prediction results for sunspot series: a. FIR($\{0, 11\}, \{0, 1\}$) with two hidden neurons, $arv = 0.115$; b. extended Elman with 2 hidden neurons, $arv = 0.165$ (prediction starts at year 1870 and is shown by a thicker line).

to avoid further increasing search complexity. A network with one hidden neuron is trained and evaluated for each configuration using NCV and then augmented by an additional neuron. This continues until a network with five hidden neurons is constructed. The lowest NCV attained is for the FIR($\{0, 11\}, \{0, 1\}$) network with two hidden neurons. For partially recurrent networks this process is repeated without the step of tap delay selection. Optimal performance is exhibited by the extended Elman network with two hidden neurons. The results of prediction for both networks are shown in Figure 2.

The extended Elman network selection process for sunspot prediction (Figure 3) can serve as an illustration of a bias-variance trade-off [10]. As the number of hidden neurons is increased, the training error decreases which shows that the model is becoming less biased and is able to represent the given data set increasingly well. At the same time the variance of the cross-validation error over a set of networks trained with different initial conditions is increasing indicating that the architecture acquires more degrees of freedom and we run a risk of overfitting. A network corresponding to the minimum of the cross-validation error NCV (two hidden neurons) is chosen as the best predictor and this assumption is confirmed by the arv having a minimum at the same network size.

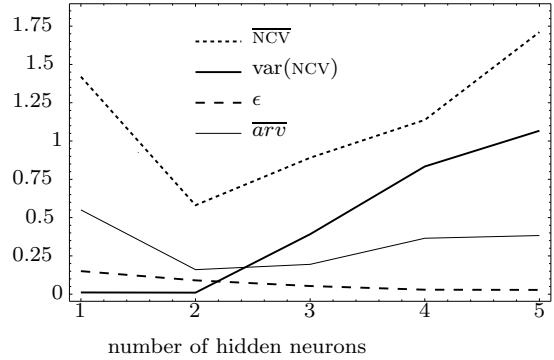


Figure 3. An illustration of bias-variance trade-off for extended Elman network trained on sunspot data (see text).

4.2. Mackey-Glass chaotic time series

This data set is a solution of the Mackey-Glass delay-differential equation

$$\dot{x}(t) = -bx(t) + \frac{ax(t - \Delta)}{1 + x(t - \Delta)^c}$$

where $\Delta = 30$, $a = 0.2$, $b = 0.1$, $c = 10$, initial conditions $x(t) = 0.9$ for $0 \leq t \leq \Delta$, and sampling rate $\tau = 6$. The training set consists of the first 500 samples. Two test sets consisting of the next 100 and 150 samples are chosen because several methods perform very well on the first standard set [8] and extending the prediction horizon makes the architecture comparison more revealing.

ARMA models have exhibited rather poor performance for this time series since the data is nonlinear. The best predictor’s output follows the oscillations for several “periods” and then slowly decays to the mean of the series.

Much better results are obtained using both Elman and FIR networks. For the first 100 steps the discrepancies between real and predicted outputs as measured by the arv_{100} index are very small. At about 120 steps and later the predictions become worse with the Elman network producing more meaningful results. Prediction error plots for the FIR and Elman networks are shown in Figure 4.

5. Conclusions

The results for different predictors are summarized in Table 1. They are valid for the methodology presented in the paper. ARMA models are tuned to produce the best results on the test set and as such, their predictions cannot be compared directly to neural network predictions which underwent a blind test on withheld data.

“Inverse pruning” in the form of the SNC algorithm, that is starting with a small architecture and then

Table 1. Best predictors in each class and their arv performance.

| Architecture | Sunsspots | | Mackey-Glass | |
|-------------------|-------------------------|-------|--------------|-----------------------|
| | size | arv | size | arv (arv_{100}) |
| ARMA [†] | AR({1, 2, 11}) | 0.252 | AR({19}) | 0.874 (0.832) |
| Elman | 3 | 0.348 | 14 | 0.214 (0.00408) |
| Extended Elman | 2 | 0.162 | 10 | 0.979 (0.133) |
| FIR | FIR({0, 11}, {0, 1}), 2 | 0.115 | FIR(6, 2), 3 | 0.447 (0.0185) |

[†] arv was measured for the validation set.

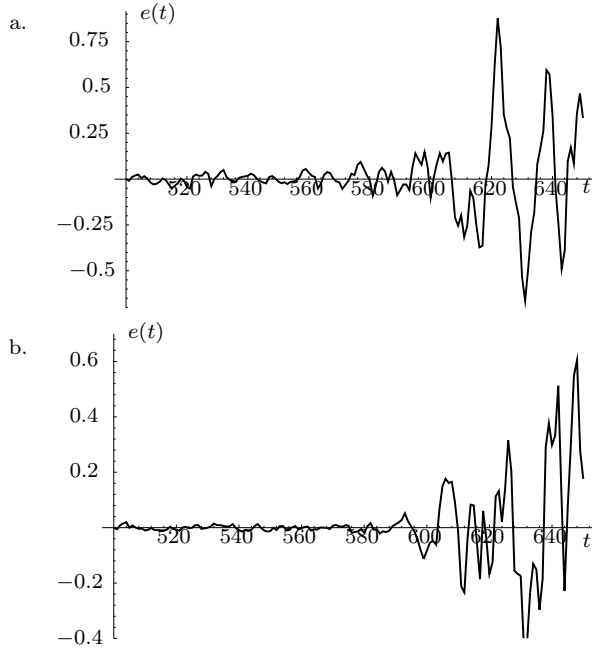


Figure 4. Prediction error plots: a. FIR(6, 2) with three hidden neurons, ($arv = 0.447$); b. Elman with 14 hidden neurons ($arv = 0.214$).

growing to incorporate more and more time series features, has been proposed in this paper. It is less computationally expensive and provides a statistical interpretation validating results for different models. Our FIR network modification enables reduction of the number of model parameters while retaining this architecture’s capability to use time lags reaching far into the past.

Possible extensions of this research include taking into account the effect of external recurrence in iterative predictions for derivative computation and the use of global minimum search techniques which should enable better predictions.

References

- [1] A. Gordon, J. P. H. Steele, and K. Rossmiller, “Intelligent engineering systems through artificial neural networks,” in *ANNIE*, (St. Louis, Missouri), ASME Press, 1991. [1.](#)
- [2] E. A. Wan, *Finite Impulse Response Neural Networks with Applications in Time Series Prediction*. PhD thesis, Stanford University, 1993. [1.](#), [2.1.](#), [2.1.](#), [2.1.](#)
- [3] A. Weigend and N. Gershenfeld, *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, 1993. Proceedings of the NATO Advanced Research Workshop on Comparative Time Series Analysis held in Santa Fe, New Mexico, May 14-17, 1992. [1.](#)
- [4] J. Moody and J. Utans, “Architecture selection strategies for neural networks: Application to corporate bond rating prediction,” in *Neural Networks in the Capital Markets*, J. Wiley & Sons, 1994. [1.](#), [3.](#), [3.](#)
- [5] M. F. Møller, “A scaled conjugate gradient algorithm for fast supervised learning,” *Neural Networks*, vol. 6, pp. 525–533, 1993. [1.](#)
- [6] A. Back, E. Wan, S. Lawrence, and A. Tsoi, “A unifying view of some training algorithms for multilayer perceptrons with FIR filter synapses,” in *Neural Networks for Signal Processing 4* (J. Vlontzos, J. Hwang, and E. Wilson, eds.), pp. 146–154, IEEE Press, 1995. [1.](#)
- [7] G. E. P. Box and F. M. Jenkins, *Time Series Analysis: Forecasting and Control*. Oakland, CA: Holden-Day, 2nd ed., 1976. [2.1.](#), [4.1.](#)
- [8] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Redwood City, California: Addison-Wesley, 1991. [2.2.](#), [4.2.](#)
- [9] R. J. Williams and J. Peng, “An efficient gradient-based algorithm for on-line training of recurrent network trajectories,” *Neural Computation*, vol. 2, pp. 490–501, 1990. [2.2.](#)
- [10] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995. [4.1.](#)