

Neural Network Tools For Stellar Light 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 presents a comparative study of state-of-the-art neurocomputing methods applied to several benchmark time series, including the white dwarf light curve. The goal is to determine which of the predictive models work best for data from natural sources. The emphasis is on using a unified methodology for selection of the best architectures among those used for comparison. The specific architectures considered are a Finite Impulse Response (FIR) network and three types of layered recurrent networks: Jordan, Elman, and extended Elman. An enhancement of a FIR network allowing selection of weights with relevant time delays only is also presented. Our approach is applied to two benchmark prediction problems: the Wölfer sunspot number data and a white dwarf light curve. Results show that the best predictions are obtained using a FIR neural network.

TABLE OF CONTENTS

1. INTRODUCTION
2. TIME SERIES MODELING ARCHITECTURES
3. NETWORK SIZE SELECTION
4. EXAMPLES AND DISCUSSION
5. CONCLUSIONS

1. INTRODUCTION

Time series prediction is one of the most important scientific tasks. Especially challeng-

ing are situations where an underlying model generating observed data is not known. Solutions in such cases can be provided by non-parametric regression methods, of which neural network (NN) based predictors are a large subclass. A number of different NN architectures and learning methods have been applied with varying degrees of success to the problem of predicting future values of various time-series. However, there still exists a need for satisfactory guidelines when choosing one approach over another for a particular problem.

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 , and the sums are taken over all T time instants in the training sequence and over all N^L outputs. The simplest method of adapting weights towards the minimum of E is the steepest descent method in which weight movement is in the direction of the negative error gradient scaled with a “learning rate” η :

$$\Delta w_{ij} = -\eta \nabla_{w_{ij}} E.$$

Weight adaptation in this paper is performed with a scaled conjugate gradient method [1] which significantly improves speed and convergence of training as compared to standard techniques. Batch mode adaptation in which

the weights are modified at the end of each training epoch is utilized. This has the property of using the exact gradient as opposed to stochastic adaptation which performs weight updates after each pattern presentation and uses a gradient approximation.

2. TIME SERIES MODELING ARCHITECTURES

ARMA models

ARMA methodology as developed by Box and Jenkins [2] dominated the field of time series analysis only until recently. An ARMA(M, N) model describes a time series $\{x_\tau\}$ as

$$x_\tau = \sum_{m=1}^M a_m x_{\tau-m} + \sum_{n=0}^N b_n \epsilon_{\tau-n} \quad (1)$$

where a_m are AR (autoregressive) coefficients, b_n are MA (moving average) coefficients, and ϵ_t values are generated by a noise source with assumed distribution. Values M and N are respectively AR and MA component orders. Example AR(2) and MA(2) predictors are shown in Figure 1.

FIR neural network

The Finite Impulse Response (FIR) network, proposed in [3] 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 in Equation (1). 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

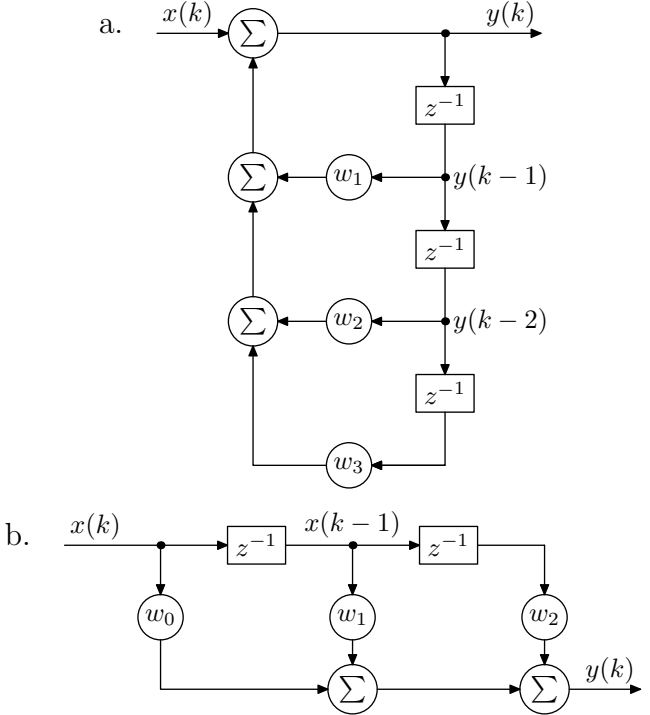


Fig. 1. Example linear predictors: a. AR(3); b. MA(2).

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

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

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

A NN structure most 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 error gradient in a FIR network is performed by means of “temporal

backpropagation” [3]:

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

where

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

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 which 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 [3] or pruning are used.

We propose a modification of a FIR NN architecture in which there are weights at desired time lags only and 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 present in the architecture while unnecessary time lags are omitted.

Partially Recurrent Neural Networks

Partially recurrent networks [4] (also known as Elman-Jordan networks) are a subclass of recurrent networks. They are multilayer perceptron networks augmented with one or more additional context layers storing output values of one of the layers delayed by one step and used for activating this or some other layer in the next time step. There are several types of these networks shown in Figure 2 which differ in the type of feedback used. The Jordan network has context

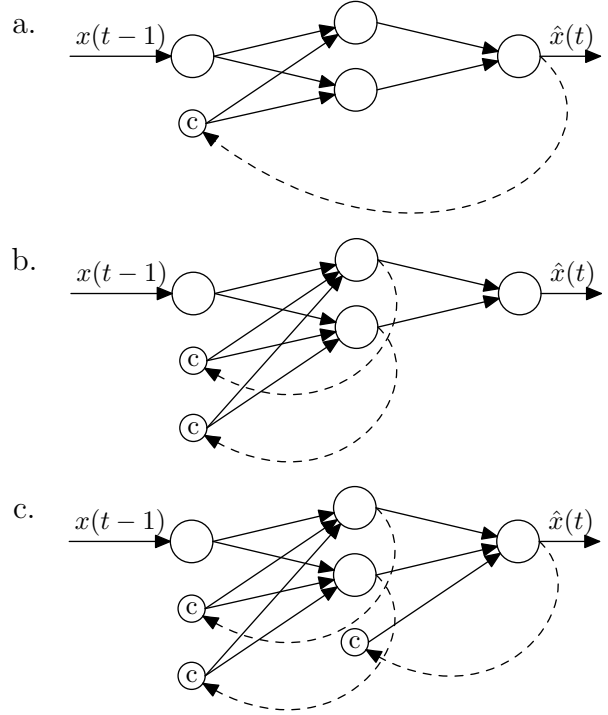


Fig. 2. Examples of partially recurrent NNs with two hidden neurons: a. Jordan; b. Elman; c. extended Elman (dashed lines represent unit delays, small circles are linear context units).

units which store delayed output values and present these as additional inputs to the network, while Elman NN has connections from a hidden layer to itself. An extended Elman network is an Elman network with an additional output-to-output feedback.

The Elman network can learn sequences that cannot be learned with the Jordan network (which is a similar architecture with a context layer fed by the output layer) since networks with only output memory cannot recall inputs that are not reflected in the output.

Several algorithms for calculation of error gradient in general recurrent networks exist. Backpropagation Through Time (BPTT) [5] is an exact and fast off-line training method. Its only relative disadvantage is substantial memory consumption as a history of network activations for all inputs has to be stored.

According to BPTT, the error gradient is calculated as follows:

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

where

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

and

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

Since architectures of Elman-Jordan networks are specific, simplified realizations of general recurrent networks, a modified version of this algorithm is used in which a pattern propagates from the input to the output layer in one time step.

3. NETWORK SIZE SELECTION

Sequential Network Construction

No general guidelines exist as to the size of NN hidden layers. This limitation leads to time consuming trial-and-error procedures. To circumvent this disadvantage we use a more systematic way of finding good architectures.

A sequential network construction (SNC) algorithm [6] is employed to select an appropriate 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 and 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.

Prediction Risk Estimation

The quality of a predictor can be defined as its ability to predict new observations. A simple way to estimate prediction error when enough 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 reuse can be applied. The best known method is cross-validation; that is, splitting the data into several disjoint subsets of roughly equal size, leaving one subset out 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.

A modification of this method for nonlinear models is Nonlinear Cross Validation (NCV) [6]. The existence of several local minima in the training error surface 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 work, the standard CV procedure can be modified by using 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: Wölfer sunspot numbers and the white dwarf light curve. 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 the subsequent predictions. Some publications describe only one-step-ahead prediction or reusing a validation set as a test set which sometimes results in artificially optimistic 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 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.

NCV is used for calculating the prediction risk using a non-iterative mode which gave better model quality estimates. Iterated predictions during validation on different data subsets exhibited a large sensitivity to weight changes.

Prediction quality is measured by means of the average relative variance

$$arv = \frac{\sum_{\tau=1}^T (x_\tau - \hat{x}_\tau)^2}{\sum_{\tau=1}^T (x_\tau - \bar{x})^2}$$

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.

All simulations on time series studied were run ten times, starting with different initial conditions. Results are averaged to reduce the influence of model variability on network size selection by introducing a possibility of escaping local minima.

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 [2] identified the optimal model for this data as AR(2); however, they also stated that this is not a particularly good fit. A lack of fit test using the first ten residual autocorrelations results in a p -value of 0.958. This is the probability that the null hypothesis that the Q quantity has an approximately chi-squared distribution with 8 degrees of freedom should be rejected and suggests that AR(2) model does not represent the data adequately. 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 significantly better results with a p -value of 0.891. Prediction for 20 years ahead gives $arv = 0.445$ and 0.252, for the above two models, respectively. Figure 3 shows the prediction results with 95% probability limits.

The best FIR network is found by using the SNC methodology for over a dozen taps configurations chosen heuristically on the basis of their performance in the ARMA model and strong periodicity of period 11 observable in the time series. Architectures considered are limited to single hidden layer networks because of their proven universal approximation capabilities and to avoid further increasing search complexity. A network with one hidden neuron is trained and tested for each configuration using NCV, then it is augmented by an additional neuron. This continues until a network with five hid-

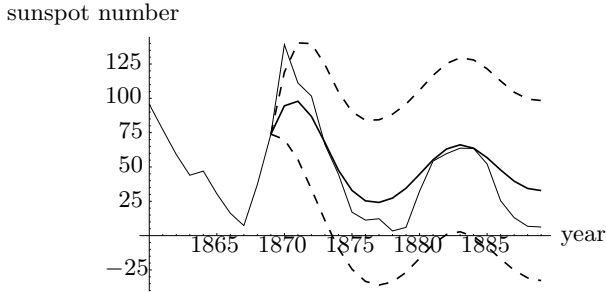


Fig. 3. Prediction results with AR model with time lags 1, 2, and 11 (prediction starts at the year 1870 and is shown by a thick line; dashed lines show 95 percent probability limits).

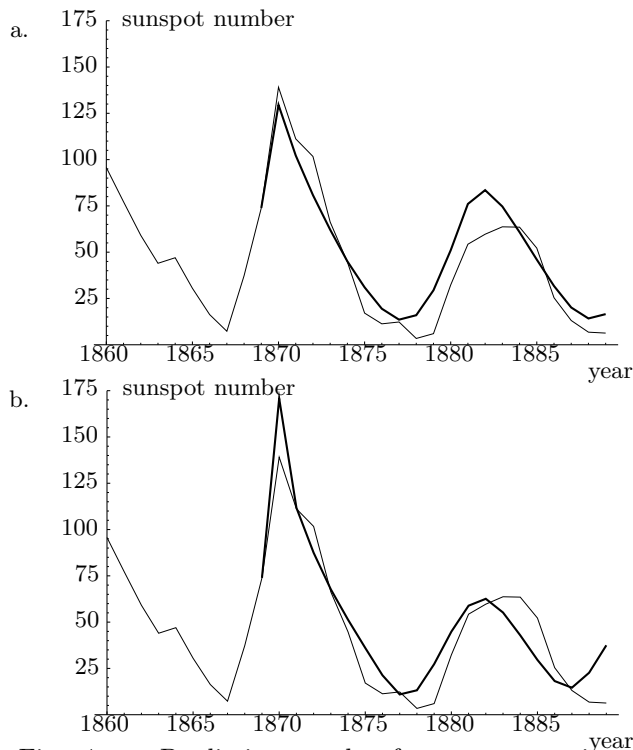


Fig. 4. Prediction results for sunspot series: a. FIR($\{0, 11\}, \{0, 1\}$) with two hidden neurons, ($arv = 0.115$); b. extended Elman with 2 hidden neurons ($arv = 0.165$).

den neurons is constructed. The lowest NCV attained is for the FIR($\{0, 11\}, \{0, 1\}$) network with two hidden neurons. For the partially recurrent networks this process is repeated without the step of tap delay selection. The best performance is exhibited by the extended Elman network with two hidden neurons. The results of prediction for both networks are shown in Figure 4.

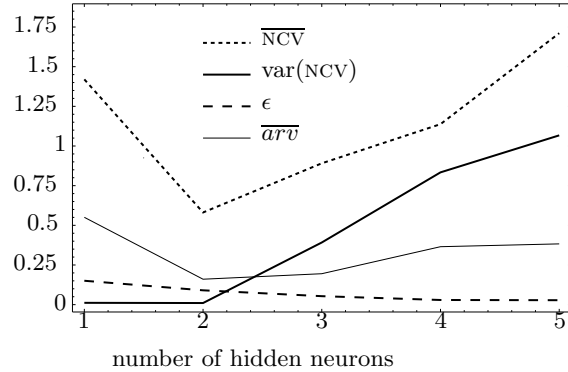


Fig. 5. An illustration of bias-variance trade-off for extended Elman network trained on sunspot data (see text).

The extended Elman network selection process for sunspot prediction (Figure 5) can serve as an illustration of a bias-variance trade-off [7]. 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.

White Dwarf Light Curve

This data set is a series of measurements of the brightness changes of the white dwarf star PG 1159-035 obtained as part of the Whole Earth Telescope Project. It was submitted as one of the data sets for the Santa Fe Institute prediction competition. It consists of 17 segments of varying lengths of which samples 200 to 999 from segment 14 are arbitrarily chosen for further analysis (mainly to avoid missing sections of data in

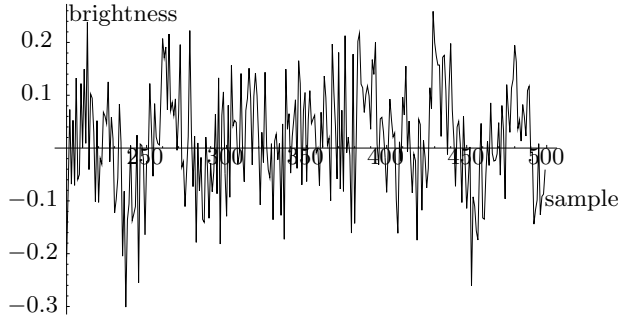


Fig. 6. A section of white dwarf light curve used in prediction experiments.

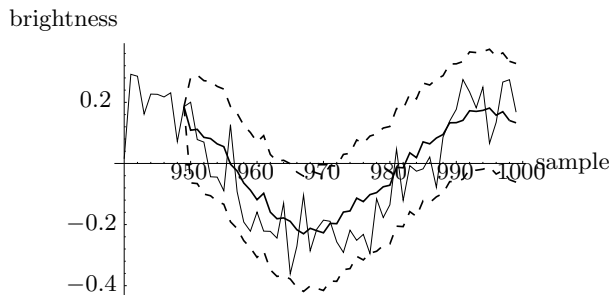


Fig. 7. Prediction results for AR model with time lags 1, 2, 48, 50, and 51 ($arv = 0.253$, predictions start at sample 950; dashed lines show 95 percent probability limits).

which linear stretches were inserted). The prediction task is to predict the samples 950 to 999 using previous samples.

This time series is very noisy (see Figure 6). Nevertheless Fourier analysis of the data reveals that there is a strong peak at 2 mHz which corresponds to a period of 50 samples. Using this time lag proves beneficial in both the ARMA model and a FIR network. This is shown in Figure 7.

We found our reduced FIR architecture to be reasonably effective (Figure 8a) in predicting of the white dwarf light curve. The best network found had hidden layer weights at lags 0 and 49 which corresponds to the shortest period. The output layer had time lags 0 and 1 which allows representation of some period instability as well. The hidden layer can be then thought of as a type of seasonal differencing operator known from lin-

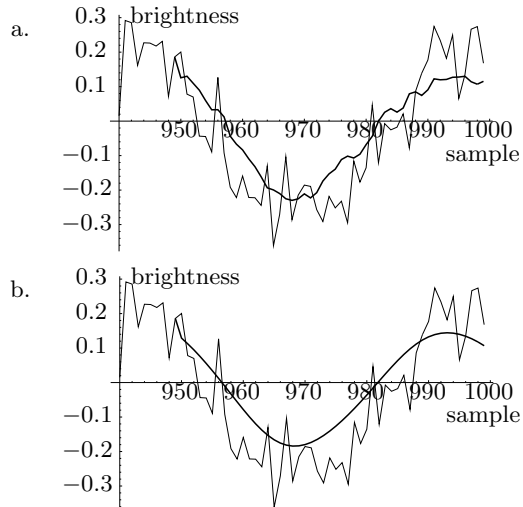


Fig. 8. Prediction results: a. FIR($\{0, 49\}, \{0, 1\}$) with one hidden neuron, ($arv = 0.287$); b. extended Elman with 3 hidden neurons ($arv = 0.325$).

ear models.

The results of extended Elman network prediction, which again performed best among recursive networks, are shown in Figure 8b.

5. CONCLUSIONS

The results for different predictors are summarized in the Table I. They are valid for the methodology explained 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. Relatively good performance of the FIR networks shows the importance of using introductory data analysis for suggesting the initial optimum lags. The fact that partially recurrent networks scored worse in this comparison can be at least partially attributed to the fact that in both cases long term memory is required on the scale of 11 or 50 time steps, and also to problems with initialization of network state using short data segments resulting from splitting the

TABLE I
BEST PREDICTORS IN EACH CLASS AND THEIR *arv* PERFORMANCE.

Architecture	Sunspots		White Dwarf	
	size	<i>arv</i>	size	<i>arv</i>
ARMA [†]	AR({1, 2, 11})	0.252	AR({1, 2, 48, 50, 51})	0.253
FIR	FIR({0, 11}, {0, 1}), 2	0.115	FIR({0, 49}, {0, 1}), 1	0.287
Jordan	1	0.575	3	1.34
Elman	3	0.348	3	0.597
Extended Elman	2	0.162	3	0.325

[†] *arv* was measured for the validation set.

data series for cross-validation purposes.

“Inverse pruning” applied here in the form of SNC algorithm, that is starting with a small architecture and then growing to incorporate more and more time series features, is less computationally expensive and provides for a good statistical interpretation. Our FIR network modification makes it possible to use the above methodology for this architecture.

Future research directions 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 allow better predictions.

REFERENCES

- [1] M. F. Møller, “A scaled conjugate gradient algorithm for fast supervised learning,” *Neural Networks*, vol. 6, pp. 525–533, 1993. **1.**
- [2] G. E. P. Box and F. M. Jenkins, *Time Series Analysis: Forecasting and Control*. Oakland, CA: Holden-Day, 2nd ed., 1976. **2., 4.**
- [3] E. A. Wan, *Finite Impulse Response Neural Networks with Applications in Time Series Prediction*. PhD thesis, Stanford University, 1993. **2., 2., 2.**
- [4] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Redwood City, California: Addison-Wesley, 1991. **2.**
- [5] 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.**

- [6] 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. **3., 3.**
- [7] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995. **4.**

Tomasz J. Cholewo received his M.Sc. degree in Electrical Engineering from the Technical University of Gdansk, Poland in 1994. He is currently a University Fellow and a Ph.D. candidate at the University of Louisville. He is an author and co-author of several conference papers in neural networks and chaotic cellular networks. His research interests include time series analysis, neuro-computing, and AI methods. He is a member of the IEEE.

Jacek M. Zurada earned his M.S. and Ph.D. from the Technical University of Gdansk, Poland. He is now the S.T. Fife Alumni Professor of Electrical Engineering at the University of Louisville, Louisville, Kentucky. He is the author of the 1992 West text “Introduction to Artificial Neural Systems,” contributor to the 1994 and 1995 Ablex volumes “Progress in Neural Networks,” and co-editor of the 1994 IEEE Press volume “Computational Intelligence: Imitating Life.” Dr. Zurada is author or

co-author of more than 120 journal and conference papers in neural networks, analog and digital VLSI circuits, and active filters. Dr. Zurada is now an Associate Editor of "IEEE Transactions on Neural Networks," of "IEEE Transactions on Circuits and Systems, Part II," "Neurocomputing," and of the "Artificial Neural Networks Journal." He is an IEEE Fellow and a Distinguished Speaker of Neural Networks Council.