Control and Cybernetics

vol. **31** (2002) No. 2

Chaotic time series prediction with feed-forward and recurrent neural nets

by

Jacek Mańdziuk[1,2] **and Rafał Mikołajczak**[1]

[1] Faculty of Mathematics and Information Science,
Warsaw University of Technology
Plac Politechniki 1, 00-661 Warsaw, Poland

[2] Corresponding author
e-mail: mandziuk@mini.pw.edu.pl

Abstract: The results of experimental comparison between several neural architectures for short-term chaotic time series prediction problem are presented. Selected feed-forward architectures (Multilayer Perceptrons) are compared with the most popular recurrent ones (Elman, extended Elman, and Jordan) on the basis of *prediction accuracy, training time requirements* and *stability*. The application domain is *logistic map series* – the well known chaotic time series predition benchmark problem.

Simulation results suggest that in terms of prediction accuracy feed-forward networks with two hidden layers are superior to other tested architectures. On the other hand feed-forward architectures are, in general, more demanding in terms of training time requirements. Results also indicate that with a careful choice of learning parameters all tested architectures tend to generate stable (repeatable) results.

Keywords: logistic map, chaotic time series, prediction, neural networks.

1. Introduction

The task of time series prediction can informally be stated as follows: given several past values of the series, predict its future value(s) within predefined prediction horizon. Depending on the horizon length the prediction tasks are generally divided into short-term and long-term predictions.

Solving the discrete time series prediction problem with the help of neural networks requires that the function defining the series be approximated with appropriate accuracy. Hence, prediction problem can be regarded as an instance of a function approximation problem. In other words, prediction problems be-as a square of prediction error in a certain future point of time or as a sum of squared prediction errors over predefined future time interval (see Section 2 for formal description).

Neural networks, due to their universal approximation properties, are widely used in financial, economical and business analyses. One of the most popular application domains in this area is financial time series prediction (Refenes, 1995; Refenes, Burgessand and Bentz, 1997). Several examples presented in the literature include prediction of the following data: stock returns (Schoneburg, 1990; Saad, Prokhorov and Wunsch II, 1998), stock market indices or volatility (S&P: Chenoweth and Obradović, 1996; Wong, 1991; Malliaris and Salchenberger, 1996; F.T.S.E.: Brownstone, 1996; TOPIX: Kohara, Fukuhara and Nakamura, 1996), or currency exchange rates (Franses and van Griensven, 1998; Kuan and Liu, 1995).

Besides the most popular applications to stock market related problems neural networks have been also successfully applied to several prediction problems, e.g. inflation rate (Zwol and Bots, 1994; Moshiri and Cameron, 2000; Moshiri, Cameron and Scuse, 1999), level of income taxes (Hansen and Nelson, 1997), unemployment rate (Moshiri, 2000), or bankruptcy risk (Leshno and Spector, 1996; Burrell and Folarin, 1997; Perez, 1999; Yang, 1999).

Another important and closely related application area is *chaotic time series prediction* often treated as an indicator of the method's quality before its application to the real (usually financial or business) data. There are two main advantages of such approach: first, chaotic series are usually less demanding in terms of complexity and time requirements, and therefore well suited for preliminary tests; second, several benchmark chaotic series (e.g. logistic map, sunspot series, Mackey-Glass equation, or Lorentz attractor) provide an adequate base for comparison between different methods.

The subject of this paper is the experimental comparison between selected feed-forward and recurrent neural architectures for short-term chaotic time series prediction based on *logistic map* series. The main issues discussed here are prediction accuracy of particular neural architectures, their training time requirements and ability to produce repeatable results of a certain quality.

Except for the above main research issues some test results are also presented for long-term predictions and for short-term ones, but with "unlimited" training time. These simulations attempted to trace the intrinsic limits of tested networks.

The work presented in this paper was partly inspired by the Hallas and Dorffner (1998) paper. In Hallas and Dorffner (1998) the authors focused on the comparative study of several feed-forward (linear and non-linear) architectures and selected recurrent ones. Similarly to our paper one of the main goals of Hallas and Dorffner (1998) was discovering *practical hints* in the area of design and application of neural networks to prediction tasks.

The paper is organized as follows: Section 2 includes introduction to chaotic

Section 3 describes preliminary experiments: neural architectures considered in the simulations (Section 3.1), training methods (Section 3.2), data sets (Section 3.4) and results (Sections 3.5 and 3.6). The main (final) experiment is presented in Section 4 along with comparison of architectures (Section 4.4) and description of additional, extended tests (Section 4.5). Final conclusions and directions for future development of this work are placed in Section 5.

2. Prediction task

The goal of time series prediction is forecasting future values given historical data of the series. Time series can formally be defined as a sequence of vectors

$$\vec{x}(t), \quad t = 0, 1, \ldots \tag{1}$$

depending on t.

Theoretically, $\vec{x}$ can be defined as a continuous function of time t, but for practical reasons it is often viewed as a sequence of values at the end points of discrete, usually fixed-size, time intervals (Dorffner, 1996).

Components of $\vec{x}$ can be any observable variables (e.g. daily level of water in the river measured at a certain hour, average monthly price of petrol, daily opening stock returns, etc.).

The above are examples of *empirical time series* in which values of $\vec{x}$ are taken from observations or measures. Another type are *artificial time series* in which $\vec{x}$ represents values of some mathematical function.

A prediction system is supposed to output value $\vec{x}(t+d)$ given last n values: $\vec{x}(t), \ldots, \vec{x}(t-n+1)$, for some n, as the input. The problem can be presented in terms of approximation task (Dorffner, 1996): find function $f : R^n \to R$ being an estimate $\widehat{\vec{x}}(t+d)$ of $\vec{x}$ at time $(t+d)$:

$$\widehat{\vec{x}}(t+d) = f(\vec{x}(t), \ldots, \vec{x}(t-n+1)), \tag{2}$$

where d is called *time lag* of prediction[1].

2.1. Chaotic time series

In our experiment deterministic time series associated with complex chaotic dynamics is considered. Logistic map time series belongs to the class of artificial series and is defined based on chaotic, recursive equations[2]. There are several mathematical models of chaotic behaviour serving as benchmarks for prediction methods. One of the simplest examples is the *simple pendulum* defined by the following equation:

$$J\ddot{\theta} + B\dot{\theta}|\dot{\theta}| + K \sin\theta = \tau, \tag{3}$$

[1]Formally, $f : R^{k\times n+l} \to R^k$, where k denotes dimension of $\vec{x}$ and l is the number of time-independent variables considered in prediction process. In our experiment k and l were fixed and equal to 1 and 0, respectively, hence they are omitted in (2).

[2]An example of *empirical* chaotic series is the *sunspot series*, i.e. the number of sunspots

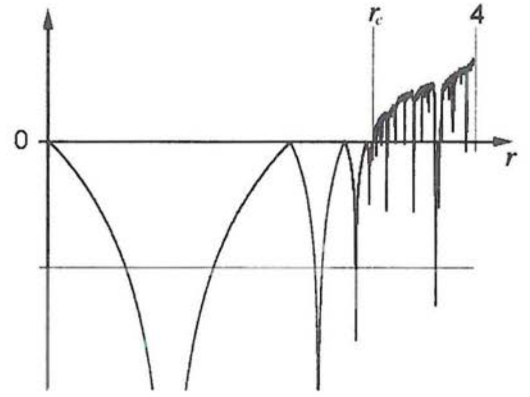


Figure 1. The plot of Lyapunov Exponent versus r for logistic map series with $x(0) = 0.2027$.

where J, B, K are appropriate coefficients (one of the choices is $J = 1, B = 0.1, K = 1$, McDonnell and Waagen, 1994).

Two more examples of benchmark artificial chaotic series include the *logistic map* (described in more detail below) and the *Mackey-Glass equation* (Mackey and Glass, 1977) representing the model for white blood cell production in leukemia patients:

$$\dot{x}(t) = \frac{ax(t-\tau)}{1 + x^c(t-\tau)} - bx(t) \tag{4}$$

where a, b, c are coefficents and τ is a time delay (e.g. $a = 0.2, b = 0.1, c = 10$ and $\tau = 30$, McDonnell and Waagen, 1994).

In the above series the intrinsic determinism in the series offers the possibilities for accurate predictions of the next few future values, but chaotic component of the system usually prevents long-term predictions (Verdes, Granitto, Navone and Ceccatto, 1998).

In this paper empirical investigation of neural networks' performance as short-term predicting models is based on logistic equation, which is defined in the following way:

$$x(t+1) = rx(t)(1 - x(t)), \quad x(0) \in (0, 1), \tag{5}$$

where $r \in \langle 1, 4 \rangle$ is a predefined parameter. The choice of r determines the "degree od chaos" in (5). For any given time series the degree of chaos in the data can be measured by the Lyapunov Exponent (Schuster, 1988; Peters, 1996). The Lyapunov Exponent (denoted by λ) is a scalar value with the following property: $\lambda > 0$ means that the series is chaotic, whereas $\lambda < 0$ characterizes

Figure 2. The first one hundred values of logistic map series generated for $r = 4$ and $x(0) = 0.2027$.

The plot of λ versus r for logistic series generated with $x(0) = 0.2027$ is presented in Fig. 1. It can be seen in the figure that deterministic behavior is observed for $r < r_c$ and chaotic one for $r_c < r \leq 4$, except for some small areas called r-windows (Schuster, 1988), where $\lambda < 0$ and the series is periodic (deterministic). The highest value of λ - which corresponds to the highest amount of chaos in logistic equation - is obtained for $r = 4$. It is interesting to note that the logistic map with $r = 4$ was previously used as a random number generator in early computers (Kugiumtzis, Lillekjendlie and Christophersen, 1995).

In order to allow for a reliable comparison with the results presented in Hallas and Dorffner (1998) the following choice of parameters was used in all experiments reported in this paper:

$$r = 4 \quad \text{and} \quad x(0) = 0.2027. \tag{6}$$

The first one hundred values of the series (5) generated with parameters (6) are presented in Fig. 2.

3. Experiment description

The experiment was organized according to the three following steps.

1. **Preliminary tests** - experimental search for optimal network design. The goal in this step was to select a set of networks for further detailed, repetitive tests. Also several combinations of training parameters (learning rate, momentum, the number of training cycles) were tested to make close to optimal choice for the main task.
2. **Detailed, repetitive tests** - the main part of the experiment. Preselected networks were trained with appropriate training parameters and tested, each in 20 trials.
3. **Additional tests** - limited number of extra tests aimed at tracing the intrinsic limits of the models. Two types of tests were performed: tests with "unlimited" number of learning cycles (only stopping condition based on increasing of validation error was applied) and tests for long-term pre-

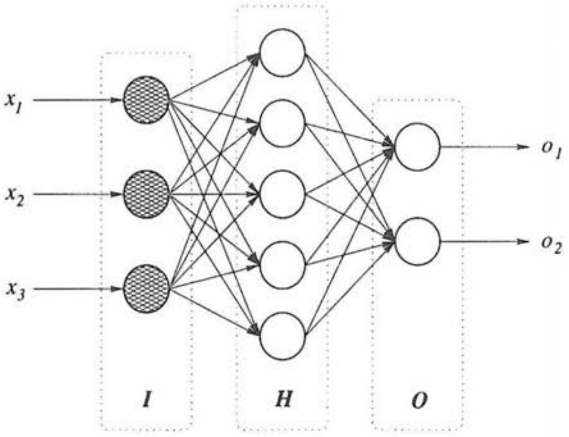


Figure 3. An example of feed-forward network with one hidden layer.

3.1. Neural network architectures

Several architectures of feed-forward and recurrent networks were compared in the experiment.

An example of *feed-forward* (f-f) network with one hidden layer is presented in Fig. 3. Networks used in the experiment were composed of sigmoidal neurons in hidden and output layers and the identity neurons in the input layer. Since f-f networks are widely used in various application domains and are well known to neural networks community, their further description here is omitted.

Recurrent networks differ from f-f ones in having feedback connections from hidden or output layers to the input one or to (additional) context layers[3]. In particular, recurrent *Jordan networks* (Jordan, 1986) (Fig. 4), considered in this paper, are one hidden layer networks with one-to-one feedback connections from the output layer to an extra context layer (therefore context and output layers have equal number of neurons). Context layer neurons are fully connected with hidden layer ones. Moreover, each context neuron has a self-recurrent (feedback) connection. Both feedback and self-recurrent connections have *a priori* defined, fixed weights which do not change during the training process. In our experiment Jordan networks were composed of sigmoidal neurons in hidden and output layers and identity ones in the input and context layers.

Another type of recurrent network was proposed by Elman (1990). Simple *Elman network* is composed of one hidden layer with extra one-to-one feedback connections to additional context layer (hence context layer has the same number of neurons as the hidden one). Moreover, according to suggestions stated

[3]Context layers are also known as *state layers*. Likewise, context neurons are also called

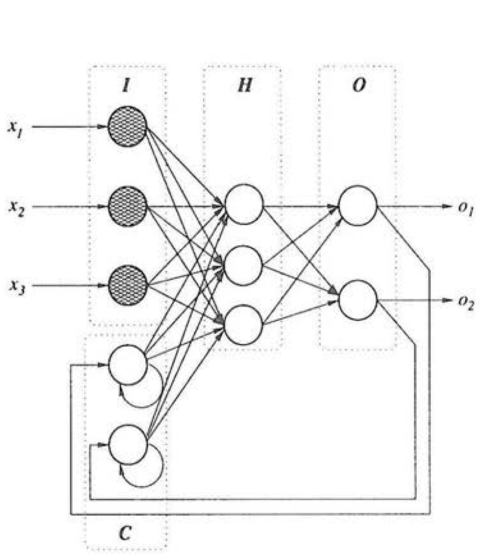


Figure 4. A Jordan neural network with a state layer.

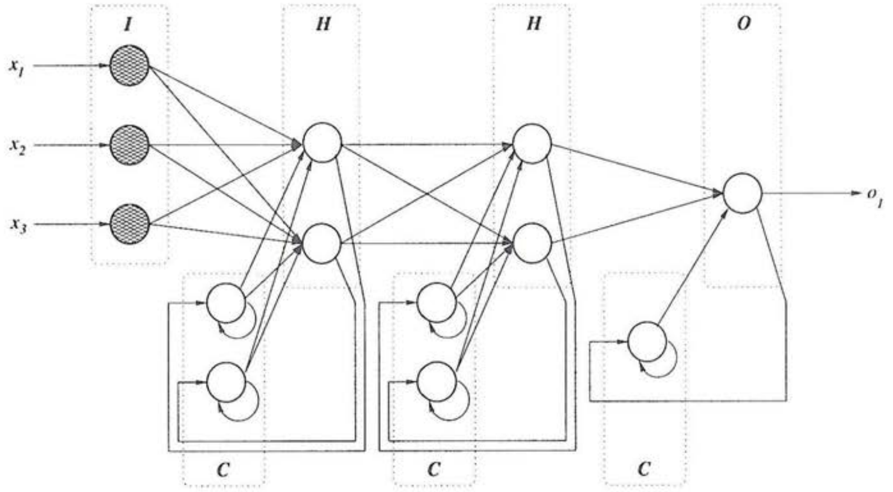


Figure 5. Extended Elman network with two hidden layers and three context layers (one contex layer per each hidden layer and additional one corresponding to the output layer).

in Pham and Xing (1995) *context units are also equipped with self-recurrent connections* – analogously to the Jordan network[4].

Extended recurrent Elman networks (Fig. 5) are composed of two or more hidden layers, each with respective context layer. Depending on the implementation there can also exist additional context layer corresponding to the output layer. Outputs of context layers are fully connected to inputs of respective hidden/output layers.

In the experiment reported in this paper both types of Elman networks (simple and extended) had feedback and self-recurrent connections with *a priori* defined, fixed weights which did not change during the training process. Both types of networks were composed of sigmoidal neurons in hidden and output layers and identity ones in the input and context layers.

3.2. Training methods

Standard backpropagation algorithm with momentum, used here for training feed-forward networks is a well known and very well documented method, see, e.g., Werbos (1974), Rumelhart, Hinton and Williams (1986a), Rumelhart, Hinton and Williams (1986b), so its description is omitted here. For recurrent networks a slightly modified version of backpropagation was used (SNNS) in whose context units were treated as if they had been additional input units (in case of Jordan and simple Elman network) or special hidden units (in case of extended Elman network). All feedback links to context units and all their self-recurrent connections had fixed, predefined weights, which remained unchanged during the learning process. In each learning epoch activation for all context units was computed and the links from context neurons to hidden or output ones were updated as if they had been links from the previous layer.

The above learning method is suitable for Jordan as well as Elman-type nets. Furthermore it can be used for any recurrent network satisfying the following restrictions (SNNS):

- input neurons are not getting input from any other neurons,
- from output neurons only links to context units are allowed,
- every unit, except for input ones, has at least one incoming link, which can be self-recurrent in case of context units,
- if all context units with their incoming and outgoing connections are removed, the remaining network becomes a multilayer perceptron[5].

Certainly, for the networks fulfilling the above criteria several other more efficient versions of backpropagation algorithm, e.g. QuickProp (Fahlman, 1988) or RProp (Riedmiller and Braun, 1992) can also be applied.

[4]Although in the original Elman's formulation (Elman, 1990) no self-recurrent connections existed, for the sake of facility of description here we shall identify Elman network as the one *with* self-recurrent connections in the context layer.

[5]Recurrent networks satisfying the above constraints (e.g. Jordan or Elman type) are also

3.3. Overlearning and overfitting

The main performance measures for neural predicting models are (Refenes, 1995):

- **convergence** – accuracy of fitting the data within the training set,
- **generalisation** – accuracy of fitting the test (out-of-sample) data,
- **stability** – low variance in prediction accuracy.

In the simulation process the data was divided into three subsets - learning, validation and test. Networks were trained on the learning set and periodically tested on validation set in order to check their generalisation abilities.

In order to achieve the best out-of-sample performance, training was stopped when validation error had increased a predefined number of times. Otherwise, if we proceeded with learning, the convergence might be still getting better but generalisation would be getting worse. Such a behaviour is a well known *over-learning* problem - the network starts to learn detailed features of the learning subset.

Similar effect of excellent convergence and poor generalisation occurs when the network's architecture is too large (compared to the size of the training set), and has therefore too many free, configurable parameters (weights). Such a situation is called *overfitting*. In our experiment the proper size of tested networks was estimated based on some theoretical hints as well as preliminary simulations.

3.4. Data sets

All simulations were performed in the Stuttgart Neural Network Simulator environment (SNNS).

Each data sample was composed of one or more input values and one output value. Based on (5) and (6) several pattern-sets were generated, each composed of 1 000 samples. Each pattern-set was divided into three subsets:

- training set - the first 700 samples;
- validation set - the next 100 samples;
- test set - the last 200 samples.

Samples, originally from the range $(0, 1)$ were scaled to the range $(0.2, 0.8)$ due to suggestion stated in (Tang and Fishwick, 1991). This suggestion was fully confirmed by preliminary tests (see Section 3.6 for details).

It is worth noting that a similar idea of "shrinking" the range of target values in supervised learning was also discussed in Gorse, Shepherd and Taylor (1997), where it is proposed to start training with the range of target values restricted to only some value – equal the average of all targets in the training set. By the use of appropriate homotopy relation the targets are then gradually "extended" to their original values. According to Gorse, Shepherd and Taylor (1997) this mechanism significantly improves network's ability to avoid local minima in the

3.5. Preliminary tests

The goal of preliminary tests was to define the suboptimal set of networks and efficient training procedures for further investigation. Since general and purely experimental approach to optimal network design can be extremly time-consuming our search was constrained to some *a priori* selected architectures. Likewise, possible sizes of hidden layers were limited to some predefined set of values.

The following features were tested in preliminary simulations:

- stability of results,
- training time requirements,
- data range: $(0.2, 0.8)$ versus $(0, 1)$,
- the maximum number of training iterations: 10 000 versus 100 000,
- different combinations of learning and momentum rates,
- adaptive algorithms for setting the learning and momentum rates,
- input size: 1-element versus 5- and 10-element inputs,
- versions of backpropagation algorithm: batch backpropagation and chunk backpropagation.

In the context of the above conditions the main conclusions drawn from preliminary tests were the following:

- 10 000 iterations was found to be enough for the purpose of qualitative comparison between neural architectures,
- networks with 1-element input provided better prediction accuracy compared to those with 5- and 10-element inputs,
- standard (batch) backpropagation with momentum outperformed other tested versions of backpropagation,
- significant improvement of results was achieved after scaling the data from $(0, 1)$ to $(0.2, 0.8)$ (see the following section for details).

3.6. Relevance of the input data scaling

One of the most important practical hints implied by this experiment is the impact of the range of data on the quality of results. Previously, Tang and Fishwick (1991) have proposed - in case of feed-forward neural nets - using the input data scaled to interval (0.2, 0.8) instead of - most commonly used - (0,1) data. This suggestion was taken into account and initially verified in our preliminary experiments *for all* tested architectures. Results of these preliminary tests are summarized in Table 1.

It can be seen from Table 1 that for all the tested architectures except only one (i.e. FF 1-30-1) the results obtained for the $(0.2, 0.8)$ interval outperformed the respective ones obtained for the $(0, 1)$ data range. In several cases the advantage was approximately equal to one order of magnitude.

Based on preliminary results a decision was made to use the data scaled to $(0.2, 0.8)$ in the main experiment. Henceforth, all references to preliminary tests

Table 1. Comparison of preliminary results for the $(0,1)$ data and the data scaled to $(0.2, 0.8)$. Each number represents the mean square error value averaged over a few trials. FF: feed-forward, EL: Elman, JO: Jordan networks. Subsequent numbers denote cardinalities of the respective layers. The values in $(0.2, 0.8)$ case presented in the table were obtained by dividing the respective original (experimental) values by 0.6.

	$(0,1)$	$(0.2,0.8)$		$(0,1)$	$(0.2,0.8)$
FF 1-20-1	$1.13E-04$	$4.87E-05$	FF 1-15-15-1	$7.68E-06$	$7.78E-07$
FF 1-30-1	$4.78E-05$	$4.98E-05$	FF 1-22-22-1	$8.38E-06$	$6.25E-07$
FF 1-60-1	$4.24E-05$	$1.14E-05$	FF 1-30-30-1	$6.00E-06$	$6.11E-07$
FF 1-100-1	$4.71E-05$	$1.05E-05$	FF 1-60-60-1	$5.62E-06$	$8.15E-07$
EL 1-10-1	$1.91E-04$	$7.10E-05$	EL 1-5-5-1	$4.92E-05$	$1.00E-05$
EL 1-20-1	$1.84E-04$	$6.80E-05$	EL 1-10-10-1	$9.75E-05$	$7.33E-06$
EL 1-30-1	$2.50E-04$	$7.43E-05$	EL 1-15-15-1	$3.84E-05$	$8.67E-06$
JO 1-10-1	$8.61E-04$	$6.67E-05$	JO 1-20-1	$7.64E-04$	$5.70E-05$
JO 1-30-1	$3.33E-04$	$5.90E-05$	JO 1-55-1	$2.74E-04$	$4.90E-05$

4. The main experiment

This section describes the main experiment. In all tests presented here, unless stated otherwise, the input layer was composed of one neuron.

4.1. Preselected architectures

The following networks were tested within the main experiment:

1. **Multilayer perceptrons** with one hidden layer of size 20, 30, 60 and 100, denoted by FF 1-20-1, FF 1-30-1, FF 1-60-1, FF 1-100-1, respectively. Preliminary tests have proved the intuitive property that the more hidden units the better prediction accuracy. On the other hand, one has to be careful with expansion of the hidden layer due to the possible overfitting.
2. **Multilayer perceptrons** with two hidden layers: FF 1-15-15-1, FF 1-22-22-1, FF 1-30-30-1 and FF 1-60-60-1. The networks had equal number of units in hidden layers, since in a set of initial tests such a selection provided for a very good performance and none of the other tested combinations performed better. In preliminary tests the network FF 1-30-30-1 slightly outperformed FF 1-60-60-1 one which suggests that for the problem considered the suboptimal cardinality of hidden layers is between 22 and 60.
3. Recurrent **Jordan networks**: JO 1-10-1, JO 1-20-1, JO 1-30-1, JO 1-55-1, with 1-element context layer. The network JO 1-55-1 allows making

4. Recurrent **Elman networks**: EL 1-10-1, EL 1-20-1 and EL 1-30-1 with 10, 20 and 30-element context layer, respectively.
5. Recurrent **extended Elman networks**: EL 1-5-5-1, EL 1-10-10-1, EL 1-15-15-1 with two context layers (of size 5, 10, 15, respectively) – one per each hidden layer. An optional context layer corresponding to the output layer was not implemented.

All weights of feedback connections in recurrent architectures (i.e. the ones from hidden/output layer neurons to context layer ones) were fixed and equal to 1. In preliminary tests weights of self-recurrent loops varied between 0.0 and 0.8. Value 0.0 (connection disabled) was found to give the best prediction accuracy and therefore in the main experiment the self-recurrent connections were disabled.

4.2. Training

The networks were trained with the following learning rates and momentum coefficients (denoted by η and α, respectively):

- Multilayer perceptrons - $\eta = 0.8, \alpha = 0.4$.
- Jordan networks - $\eta = 0.15, \alpha = 0.05$.
- Elman networks - $\eta = 0.2, \alpha = 0.05$.
- In case of extended Elman networks we were unable to select fixed η and α which would provide stable and efficient results. Therefore, simple adaptive algorithm (**Algorithm 1** presented below) was used for changing learning rate and momentum for this type of networks.

Algorithm 1 [*Adaptive selection of learning rate and momentum for extended Elman networks*]

1. For the first 200 epochs set relatively small rates: $\eta = 0.1$ and $\alpha = 0.05$ to allow soft preorientation of units and weights.
2. After the above initial period set high values for learning coefficients: $\eta = 0.8$ and $\alpha = 0.4$.
3. During the training process, each time a certain (predefined) level of the *mean square error* is reached, divide η and α by 2.

Each of 18 networks was trained in 20 different runs. Validation was performed every 200 epochs. Training was stopped either after 10 000 epochs or after three subsequent increments of validation error.

4.3. Results

The training procedure defined in Section 4.2 allows the learning process to be stopped before reaching the limit for a maximum number of epochs. In some simulations the learning process became stuck in a local minimum at a very early stage (e.g. after several hundred epochs). Consequently, prediction accuracy in took less than 1 500 epochs (15% of a possible maximum) were not taken into account in the final comparison[6]. The number of discarded (unsuccessful) tests varied from 0 for feed-forward networks up to 4 for the worst case of recurrent ones.

For the sake of comparisons with literature all results obtained for the data range $(0.2, 0.8)$ were scaled back to correspond to the range $(0, 1)$. Rescaled results for all tested architectures are presented in Fig. 6 (multilayer perceptrons), Fig. 7 (Jordan networks) and Fig. 8 (Elman networks). In the figures $MEAN$ denotes the average MSE (*mean square error*) on the test set calculated for all successful tests, $STD.DEV.$ denotes the *standard deviation* calculated as:

$$STD.DEV. = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(MSE_i - MEAN)^2}, \tag{7}$$

where n is the number of successful tests and MSE_i is the *mean square error in test* i. $STD.ERR.$ denotes the standard estimation error (Luszniewicz and Słaby, 1997) calculated as:

$$STD.ERR. = \frac{STD.DEV.}{\sqrt{n-1}}. \tag{8}$$

The main reason for using several network architectures of the same type (differing by hidden layers' size only) was the search for the best out-of-sample prediction accuracy, which means the best generalisation ability. Therefore, it is reasonable to expect that results should be poor for networks too small for our problem, but also for too large ones (due to the overfitting problem) - with suboptimal results achieved somewhere in between.

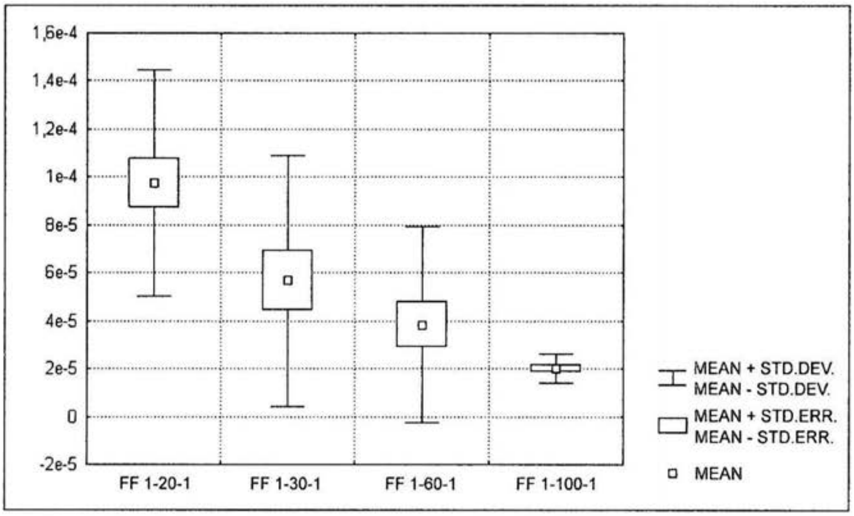
For multilayer perceptrons (MLPs) presented in Fig. 6 overfitting was observed only in case of the nets with two hidden layers. Based on the results we conclude that FF 1-30-30-1 is close to the optimal architecture among MLPs with two hidden layers since results for FF 1-60-60-1 architecture are visibly worse.

In case of MLPs with one hidden layer overfitting was not observed – the best performance on the test sets was achieved by the largest network FF 1-100-1. Probably, in order to observe the overfitting phenomenon, the size of the hidden layer should be extended beyond one hundred.

Overfitting was not observed for Jordan nets, either. As can be seen in Fig. 7 the network JO 1-20-1 performed better than JO 1-10-1 and JO 1-30-1, but worse than JO 1-55-1. Hence, the results for overfitting test are inconclusive and this issue needs further and more detailed investigation. It is worth noting that results for the JO 1-55-1 net were nevertheless almost three times smaller than those presented in Hallas and Dorffner (1998) for the same architecture,

[6]In effect for some architectures the number of relevant final tests was actually less than

(a)



(b)

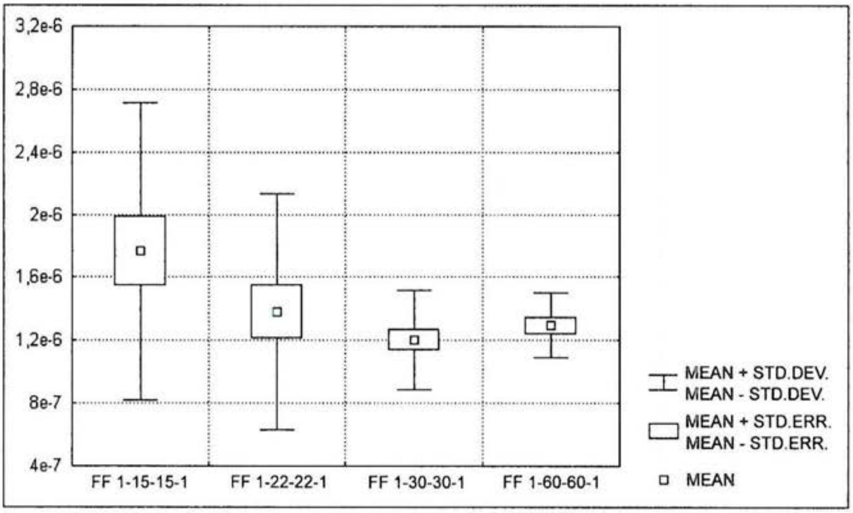


Figure 6. Prediction accuracy of multilayer perceptrons on test data sets – see description in the text.

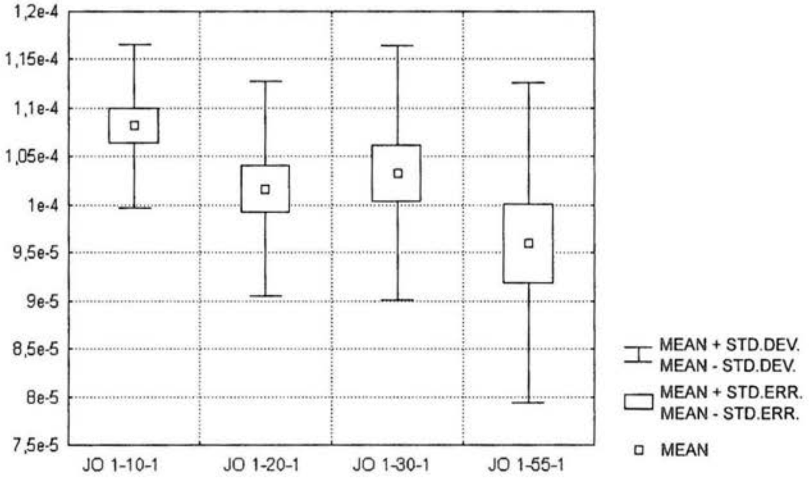


Figure 7. Prediction accuracy of Jordan networks on test data sets – see description in the text.

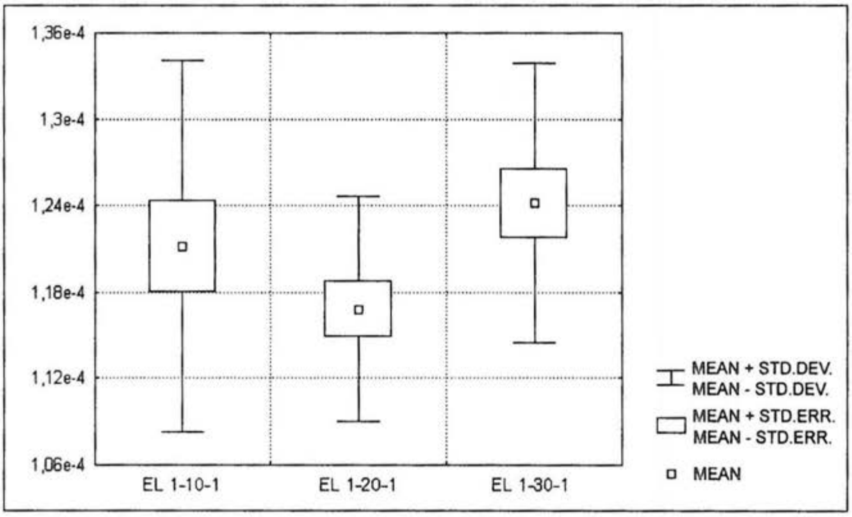
the same chaotic series and the same limits for the maximum number of epochs. This significant difference is probably caused by scaling the data to the range $(0.2, 0.8)$ and by using backpropagation with momentum, contrary to Hallas and Dorffner (1998), where data from the range $(0, 1)$ were considered and the learning algorithm was standard backpropagation.

In both types of Elman nets (Fig. 8) overfitting was clearly observed. Results suggest that Elman architectures EL 1-20-1 and EL 1-10-10-1 are close to optimal within the respective types of architectures. It is worth recalling here that the extended Elman networks were the only ones for which the learning rate and the momentum were selected in an adaptive manner (i.e. were not fixed).

4.4. Comparison of network architectures

Comparison of the winning architectures in each network type is shown in Fig. 9 and in more detail in Table 2. The main conclusions drawn from these comparative results are the following:

- The best out-of-sample predictions were obtained for feed-forward networks with two hidden layers. The winning architecture was FF 1-30-30-1 with $MEAN \approx 1.2 \cdot 10^{-6}$.
- Very good results, but one order of magnitude worse than for 2-hidden layer MLPs, were obtained for extended Elman architectures with adaptive selection of learning rate and momentum. These results were in turn one order of magnitude better than those for Jordan and Elman architectures

(a)


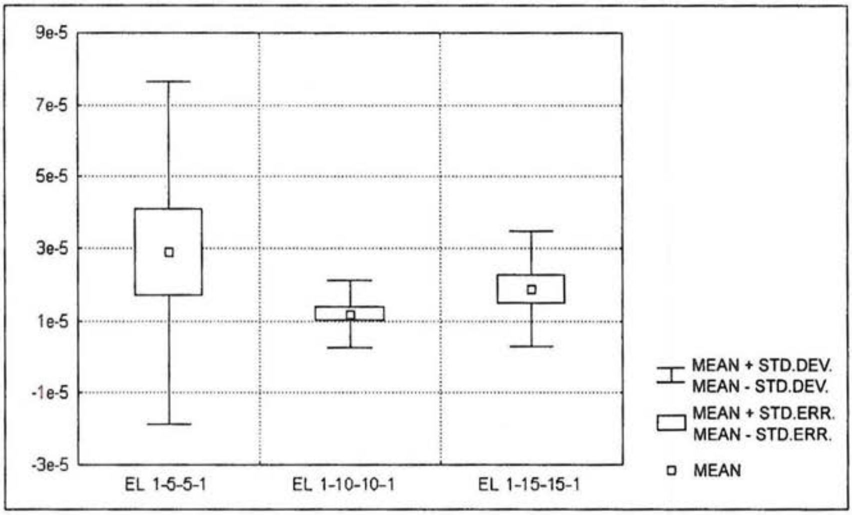
(b)


Figure 8. Prediction accuracy of simple and extended Elman networks on test data sets – see description in the text.

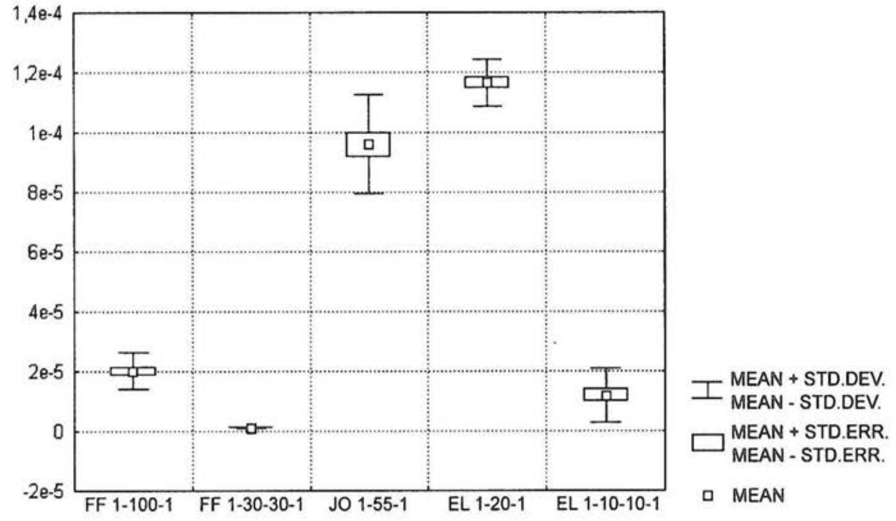


Figure 9. Prediction accuracy - comparison of the best architectures in each of five types of networks.

Table 2. Comparison of five winning architectures - one per each network type. MSE_L and MSE_T denote errors on learning and test data, respectively. $MEAN$, $STD.DEV.$, MIN denote the mean value, standard deviation and minimum of the respective MSE among all successful runs.

	MSE_L		
	$MEAN$	$STD.DEV.$	MIN
FF 1-100-1	$1.66E-05$	$4.97E-06$	$6.97E-06$
FF 1-30-30-1	$1.18E-06$	$3.17E-07$	$5.93E-07$
JO 1-55-1	$8.85E-05$	$1.20E-05$	$7.30E-05$
EL 1-20-1	$1.02E-04$	$5.75E-06$	$8.96E-05$
EL 1-10-10-1	$9.41E-06$	$6.88E-06$	$3.32E-06$
	MSE_T		
	$MEAN$	$STD.DEV.$	MIN
FF 1-100-1	$2.01E-05$	$6.20E-06$	$8.41E-06$
FF 1-30-30-1	$1.20E-06$	$3.17E-07$	$6.02E-07$
JO 1-55-1	$9.60E-05$	$1.66E-05$	$7.64E-05$
EL 1-20-1	$1.17E-04$	$7.83E-06$	$1.05E-04$
EL 1-10-10-1	$1.19E-05$	$9.19E-06$	$4.98E-06$

- The best result among perceptrons with 1-hidden layer was 1.5 order of magnitude worse than the best result for 2-hidden layer perceptrons. Based on the results it is suggested that networks with 2-hidden layers (both feed-forward and recurrent) are more effective in short-term prediction task of chaotic time series than their 1-hidden layer counterparts.
- Recurrent networks converged faster than feed-forward ones, but on the other hand were generally more sensitive to changes of learning and momentum rates.
- Significant improvement was achieved by scaling the data from interval $(0, 1)$ to $(0.2, 0.8)$.
- For all architectures the results were repeatable with relatively low variances.

Our results, for all tested types of architectures were superior to these presented in Hallas and Dorffner (1998), where a similar experiment was reported. Seeking for possible explanations of these differences we exactly repeated Hallas and Dorffner's tests and obtained results very close to theirs. Further analysis of both experiments led to three reasons, which most probably caused superiority of our results:

- first, in our experiment learning with momentum was applied,
- second, the data was scaled to $(0.2, 0.8)$[7],
- third, individual learning and momentum coefficients (or adaptive scheme in case of extended Elman networks) were applied in our tests, whereas in Hallas and Dorffner (1998) the same coefficients were used for all types of networks.

4.5. Extended tests

After the main experiment was concluded some additional tests were performed aiming at

1. checking the limits of prediction accuracy of the best MLP architectures,
2. evaluation of **Algorithm 1** on other types of networks,
3. checking the ability of tested networks to perform long-term predictions,
4. more accurate performance comparison in case of $(0.2, 0.8)$ data versus $(0, 1)$ data for the five winning architectures.

Ad. 1. Training with unlimited number of epochs (only stopping condition based on validation error incrementation was applied) was performed for multilayer perceptrons only, since in recurrent networks training was usually stopped before reaching the maximum number of epochs. The two best MLP architectures: FF 1-30-30-1 and FF 1-60-60-1 were considered. Training parameters were the same as in the main experiment and the test was performed twice for each net. Results are presented in Table 3. In each test the number of training epochs until the stopping condition was satisfied was aproximately equal to

Table 3. Results of the test with "unlimited" number of epochs. MSE_L, MSE_V and MSE_T denote learning, validation and test errors, respectively. $MEAN$ denotes the mean value of MSE_T for two tests.

architecture	FF 1-30-30-1		FF 1-60-60-1	
error	test 1	test 2	test 1	test 2
MSE_L	$3.33E-8$	$3.08E-8$	$4.54E-8$	$3.27E-8$
MSE_V	$3.75E-8$	$2.85E-8$	$4.74E-8$	$3.46E-8$
MSE_T	$3.70E-8$	$3.43E-8$	$4.93E-8$	$3.66E-8$
$MEAN$	$3.56E-8$		$4.29E-8$	

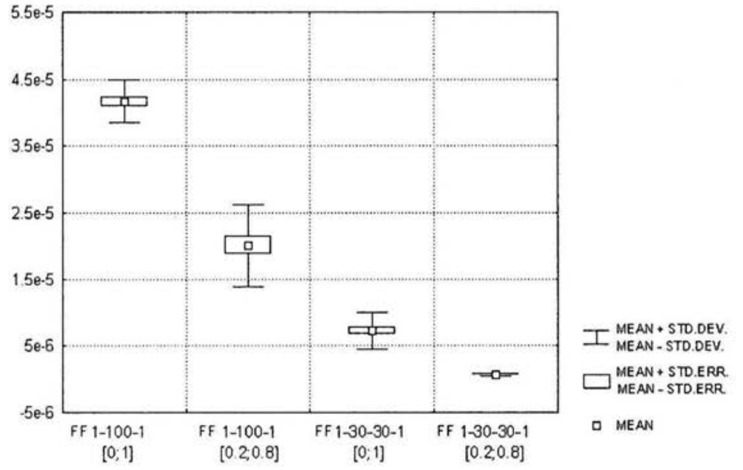


Figure 10. Comparison of results achieved by the best feed-forward architectures for the $(0,1)$ and $(0.2,0.8)$ data ranges.

200 000. The results confirm the claim that architecture FF 1-30-30-1 is close to optimal among 2-hidden layer perceptrons and that worse predictions output from the FF 1-60-60-1 net were caused by overfitting rather than by too short training time.

Ad. 2. Application of the adaptive scheme for learning rate and momentum (**Algorithm 1**) in the Jordan and simple Elman networks resulted in performance comparable to that achieved with fixed, predefined rates.

Ad. 3. The test for long-term prediction was performed for two architectures: the winning MLP, i.e. FF 1-30-30-1 and its variant with extended input layer, i.e. FF 10-30-30-1. Both networks were tested with the time lag $d=2,3,4,5$.

Both architectures performed rather poorly and FF 10-30-30-1 was generally

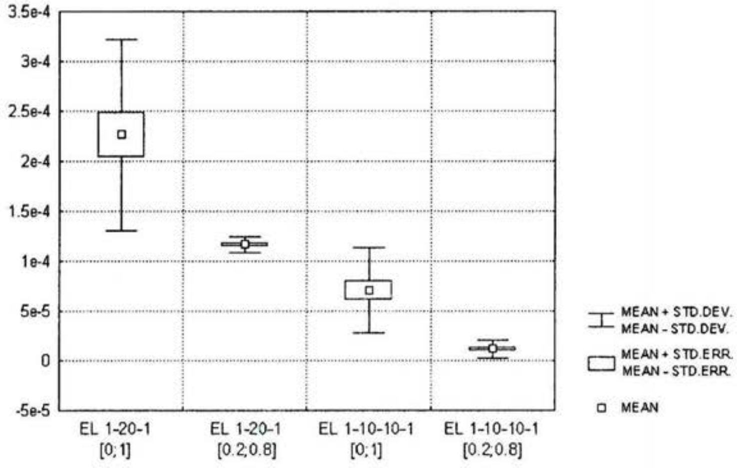


Figure 11. Comparison of results achieved by the best Elman architectures for the $(0, 1)$ and $(0.2, 0.8)$ data ranges.

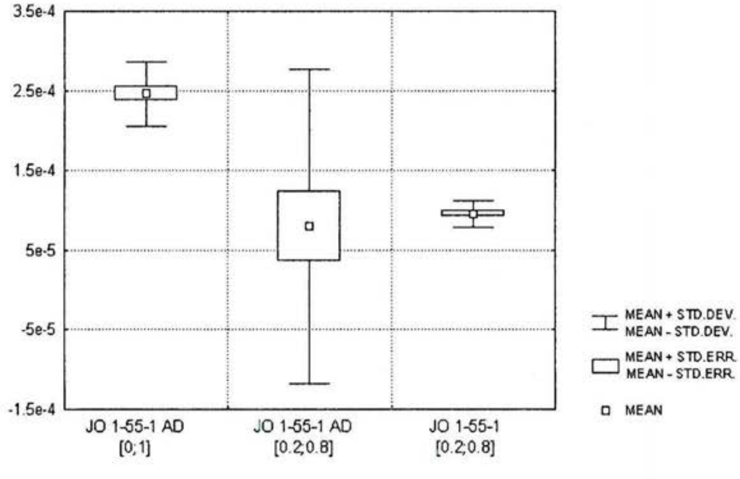


Figure 12. Comparison of results achieved by the best Jordan architecture for the $(0, 1)$ and $(0.2, 0.8)$ data ranges. AD denotes *adaptive* learning scheme.

increasing $1.5 - 2$ orders of magnitude each time the time lag was incremented by 1, until $d = 4$. Practically, predictions for $d \geq 4$ were useless. The actual values of MSE_T were the following: $9E - 06$ for $d = 2$, $5.8E - 04$ for $d = 3$, $4.6E - 02$ for $d = 4$ and $4.7E - 02$ for $d = 5$.

Ad. 4. In preliminary tests it was observed that scaling the data to interval $(0.2, 0.8)$ resulted in evident improvement of performance for almost all tested architectures. Since this observation seems to be relevant (both theoretically and practically) additional tests for $(0, 1)$ data were performed in order to compare them with results from the main experiment. Comparative results presented in Figs. 10, 11 and 12 (for feed-forward, Elman and Jordan nets, respectively) ultimately confirmed our preliminary hypothesis about "superiority" of the $(0.2, 0.8)$ data range.

5. Final conclusions and directions for future work

Results presented in this paper indicate that neural nets with two hidden layers are superior to those with one hidden layer in short-term predictions of chaotic time series. It is also shown that contrary to the common beliefs feed-forward nets may be better suited for this task than recurrent ones. An interesting observation is high stability of results for all tested architectures.

The above conclusions may, on the other hand, be appropriate only for a certain class of chaotic series and not necessarily in general case. The reason for such a remark is the fact that the logistic map recursive equation is defined based on the last time step (one step back) only, and therefore the inherent recursive mechanism in recurrent neural architectures may possibly not be fully explored in that case. In order to verify this conjecture similar experiments are planned for other chaotic series, which are defined on the basis of several (more than one) past values.

One of the practical hints implied by this work is visible improvement of results caused by scaling the data to the range $(0.2, 0.8)$ instead of $(0, 1)$. This issue, previously reported in Tang and Fishwick (1991), deserves further theoretical and experimental investigation. Another interesting issue for future research is verification of qualitative conclusions presented in this paper on financial or business data.

References

BROWNSTONE, D. (1996) Using percentage accuracy to measure neural network predictions in Stock Market movements. *Neurocomputing*, **10**, 237–250.

BURRELL, P.R. and FOLARIN, B.O. (1997) The impact of neural networks in finance. *Neural Computing & Applications*, **6**, 193–200.

CHENOWETH, T. and OBRADOVIĆ, Z. (1996) A multi-component nonlinear

DORFFNER, G. (1996) Neural Networks for Time Series Processing. *Neural Network World*, **6** (4), 447–468.

ELMAN, J.L. (1990) Finding Structure in Time. *Cognitive Science*, **14**, 179–211.

FAHLMAN, S.E. (1988) Faster learning variations on backpropagation: an empirical study. *Proc. 1988 Connectionist Models Summer School*, Morgan Kaufmann, Los Altos, USA, 38–51.

FRANSES, P.H. and GRIENSVEN, K. (1998) Forecasting exchange rates using neural networks for technical trading rules. *Studies in Nonlinear Dynamics and Econometrics*, **2** (4), 109–114.

GORSE, D., SHEPHERD, A.J. and TAYLOR, J.G. (1997) The new ERA in supervised learning. *Neural Networks*, **10** (2), 343–352.

HALLAS, M. and DORFFNER, G. (1998) A comparative study on feedforward and recurrent neural networks in time series prediction using gradient descent learning. In: Trappl, R., ed., *Proc. of the 14th European Meeting on Cybernetics and Systems Research, Austrian Society for Cybernetics Studies*, Vienna, **2**, 644-647.

HANSEN, J.V. and NELSON, R.D. (1997) Neural Networks and Traditional Time Series Methods: A Synergistic Combination in State Economic Forecasts. *IEEE Transactions on Neural Networks*, **8** (4), 863–873.

JORDAN, M.I. (1986) Serial order: A parallel distributed processing approach. *Technical Report 8604*, Institute for Cognitive Science, University of California, San Diego, La Jolla, CA, USA.

KOHARA, K., FUKUHARA, Y. and NAKAMURA, Y. (1996) Selective learning for neural network forecasting of stock markets. *Neural Computing & Appl.*, **4**, 143–148.

KUAN, C.M. and LIU, T. (1995) Forecasting exchange rates using feed-forward and recurrent neural networks. *Journal of Applied Econometrics*, **10**, 347–364.

KUGIUMTZIS, D., LILLEKJENDLIE, B. and CHRISTOPHERSEN, N. (1995) Chaotic Time Series Part I: Estimation of some invariant properties in state space. *Modelling, Identification and Control*, **15(4)**, 205–224.

LESHNO, M. and SPECTOR, Y. (1996) Neural network prediction analysis: The bankruptcy case. *Neurocomputing*, **10** (2–4), 125–147.

LUSZNIEWICZ, A. and SŁABY, T. (1997) *Applied Statistics*. PWE (in Polish).

MACKEY, M.C., and GLASS, L. (1977) Oscillation and chaos in physiological control systems. *Science*, **197**.

MALLIARIS, M. and SALCHENBERGER, L. (1996) Using neural networks to forecast the S& P 100 implied violatility. *Neurocomputing*, **10**, 183–195.

MCDONNELL, J.R. and WAAGEN, D. (1994) Evolving recurrent perceptrons for time-series modeling. *IEEE Transactions on Neural Networks*, **5** (1), 24–38.

MOSHIRI, S. (2000) Forecasting asymmetric unemployment rates. A comparison between linear, non-linear, and ANN models. *Proc. of* 20th *Int. Symposium of Forecasting*, Lisbon.

Moshiri, S. and Cameron, N. (2000) Econometrics vs. ANN models in forecasting inflation. *Journal of Forecasting*, **19**, 201–217.

Moshiri, S., Cameron, N. and Scuse, D. (1997) Static, dynamic and hybrid neural networks in forecasting. *Computational Economics*, **14**, 219–235.

Perez, M. (1999) Neural networks applications in bankraptcy forecasting: a state of the art. *Proc. of the European Symposium on Intelligent Techniques*, Greece.

Peters, E.E. (1996) *Chaos and Order in the Capital Markets. A New View of Cycle, Prices and Market Volatility.* John Wiley & Son.

Pham, D. and Xing, L. (1995) Dynamic system identification using Elman and Jordan networks. In: Bulsari, A., ed., *Neural Networks for Chemical Engineering*, **chap. 23**, 572–591.

Refenes, A-P. (1995) *Neural Networks in the Capital Markets.* John Wiley & Sons.

Refenes, A-P.N., Burgess, A.N. and Bentz, Y. (1997) Neural Networks in Financial Engineering: A Study in Methodology. *IEEE Trans. on Neural Networks*, **8**, 1222–1267.

Riedmiller, M. and Braun, H. (1992) RPROP - a fast adaptive learning algorithm. *Technical Report*, University of Karlsruhe.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986a) Learning Representations by Back-Propagating Errors. *Nature*, **323**, 533–536.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986b) Learning Internal Representations by Error Propagation. In: Rumelhart, D.E. and McClelland, J.L., eds., *Parallel Distributed Processing*, **1**, chap. 8.

Saad, E.W., Prokhorov, D.C. and Wunsch II, D.C. (1998) Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Trans. on Neural Networks*, **9** (6), 1456–1470.

Schoneburg, E. (1990) Stock price prediction using neural networks: A project report. *Neurocomputing*, **2**, 17–27.

Schuster, H.G. (1988) *Deterministic Chaos. An Introduction.* VCH Weinheim.

SNNS Stuttgart Neural Network Simulator – Manual. *http://ra.informatik.uni-tuebingen.de/SNNS.*

Tang, Z. and Fishwick, P.A. (1991) Feed-forward neural nets as models for time series forecasting. *Technical Report tr91-008*, University of Florida, USA.

Verdes, P.F., Granitto, P.M., Navone, H.D. and Ceccatto, H.A. (1998) Forecasting chaotic time series: Global vs. local methods. In: Pfeifer, J., ed., *Novel Intelligent Automation and Control Systems*, **1**, ALFA-NIACS, 129–145.

Weigned, A.S., Huberman, B.A. and Rumelhart, D.E. (1990) Predicting the future: a connectionist approach. *International Journal of Neural Systems*, **1** (3), 193–209.

WERBOS, P.J. (1974) *Beyond regression: new tools for prediction and analysis in the behavioral sciences.* Ph.D. thesis, Harvard University, Cambridge, MA, USA.

WONG, F.S. (1991) Time series forecasting using backpropagation neural networks, *Neurocomputing*, **2**, 147–159.

YANG, Z.R. (1999) Probabilistic neural networks in bankruptcy prediction. *Journal of Business Research*, **44**, 67–74.

ZWOL, W. and BOTS, A. (1994) Experiments with neural networks: Forecasting the German inflation rate. *Proc. Int. Conf. on Artificial Neural Networks – ICANN'94*, **II**, 879–882.