Dynamic Neural Network for Prediction and Identification of Nonlinear Dynamic Systems

Dubravko Majetić

Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, Zagreb, Croatia

An attempt has been made to establish a nonlinear dynamic discrete-time neuron model, the so called Dynamic Elementary Processor (DEP). This dynamic neuron disposes of local memory, in that it has dynamic states. Based on the DEP neuron, a Dynamic Multi Layer Perceptron Neural Network is proposed to predict a time series of nonlinear chaotic system. As an another application of the proposed Dynamic Neural Network (DNN), the identification of a dynamic discrete-time nonlinear system whose measurement data are spoiled with noise is performed. To accelerate the convergence of proposed extended dynamic error back propagation learning algorithm, the momentum method is applied. The learning results are presented in terms that are insensitive to the learning data range and allow easy comparison with other learning algorithms, independent of machine architecture or simulator implementation.

Keywords: discrete dynamic neuron model, dynamic error-back propagation, nonlinear signal processing, chaotic system prediction, nonlinear system identification

1. Introduction

Since artificial neural networks can effectively represent complex nonlinear functions, they proved to be a very useful tool in prediction and identifying of highly nonlinear systems. The neuron models most commonly applied are the Feed Forward Perceptron used in multi layer networks, and the Radial Basis Function neuron (RBF). Both networks have proved to be universal approximator of any static nonlinear mapping [Ni90]. They are capable of identifying any nonlinear unique state function to arbitrary desired accuracy. Recently, interests have been increasing towards the usage of neural networks for modeling and identification of dynamic systems. These networks, naturally, involve dynamics in the form of feedback connections and are known as Recurrent Neural Networks. Several learning methods for recurrent networks have been proposed in literature. Most of these methods rely on the gradient methodology and involve the computation of partial derivatives, or sensitivity functions. In this sense, they are extension of the well known error-back propagation algorithm for feedforward neural networks [Zu92].

Examples of such learning algorithms [Na90], [Ko92] include the error-back propagation through-time algorithms, the real-time recurrent learning algorithm, the recurrent backpropagation, and dynamic backpropagation.

Theoretical works by several researchers, including Cybenko [Cy89] and Funahashi [Fu89], have proven that, even with one hidden layer, artificial neural networks can uniformly approximate any continuous function over a compact domain, provided the network has sufficient number of units, or neurons.

Thus the network proposed in this study and plotted in Fig. 2 has three layers. Each i-th neuron in the first, input layer has single input which represents the external input to the neural network. The second layer, which has no direct connections to the external world, is usually referred to as a hidden layer consisting of dynamic neurons which are presented by Fig. 1. Each j-th dynamic neuron in hidden layer has an input from every neuron in the first layer, and one additional input with a fixed value of unity usually named as Bias. Each k-th neuron in the third, output layer has an input from every neuron in the second layer and, like the second layer, one additional input with fixed value of unity (Bias). The output of the third layer is the external output of the neural network.

For proposed neural network with dynamic neurons in hidden layer, the extended error-back propagation supervised learning algorithm is developed and neurons weights and coefficients of dynamic neurons are learned simultaneously. In this work the learning and generalization capability of developed learning algorithm are tested in two examples. The first example is prediction of nonlinear chaotic system presented by Glass–Mackey equation. The second example of possible usage of proposed neural network structure is the identification of a dynamic nonlinear system.

2. Dynamic Neuron Model

The basic idea of the dynamic neuron concept is to introduce some dynamics to the neuron transfer function, such that the neuron activity depends on the internal neuron states. In this study an ARMA (Auto Regressive Moving Average) filter is integrated within the well known static neuron model. Such a filter allows the neuron to act like an infinite impulse response filter, and the neuron processes past values of its own activity an input signals. The structure of a proposed dynamic neuron model, the so called Dynamic Elementary Processor (DEP), is plotted in Fig. 1.

The filter input and output at time instant (n)

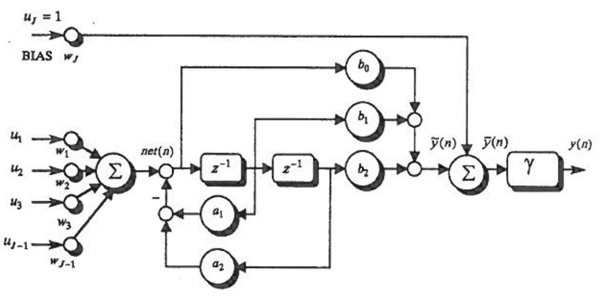

Fig. 1. Dynamic neuron model

are given in (1) and (2) respectively:

$$\mathrm{net}(n) = \sum_{j=1}^{J-1} w_j u_j \tag{1}$$

$$\tilde{y}(n) = b_0\,\mathrm{net}(n) + b_1\,\mathrm{net}(n-1) + b_2\,\mathrm{net}(n-2) - a_1\tilde{y}(n-1) - a_2\tilde{y}(n-2) \tag{2}$$

The input of the neuron activation function is given in (3), and nonlinear continuous bipolar activation function is described in (4):

$$\bar{y}(n) = \tilde{y}(n) + w_J u_J \tag{3}$$

where $u_J = 1$ represents a threshold unit, also called Bias.

$$y(n) = \gamma(\bar{y}(n)) = \frac{2}{1 + e^{-\bar{y}(n)}} - 1 \tag{4}$$

3. Learning Algorithm for Optimal Parameters

The goal of the learning algorithm is to adjust the neural network parameters (both the weights and filter coefficients) based on a given set of input and desired output pairs (supervised learning) and to determine the optimal parameter set that minimizes a performance index E as follows:

$$E = \frac{1}{2}\sum_{n=1}^{N}(O_d(n) - O(n))^2 \tag{5}$$

where N is the training set size, and the error is the signal defined as difference between the desired response $O_d(n)$ and the actual neuron response $O(n)$. This error, which is calculated at the output layer (Fig. 2), is propagated back to the input layer through the dynamic filters of dynamic neurons in hidden layer. The result is an extended, dynamic error-back propagation learning algorithm. The adjustment of weights and filter coefficients occurs for each input-output data pair (pattern or stochastic learning procedure).

The linear activation function given in (6) is a chosen transfer (activation) function for static neurons in the output layer.

$$O_k(n) = \gamma_k(\mathrm{net}_k(n)) = \mathrm{net}_k(n) \tag{6}$$

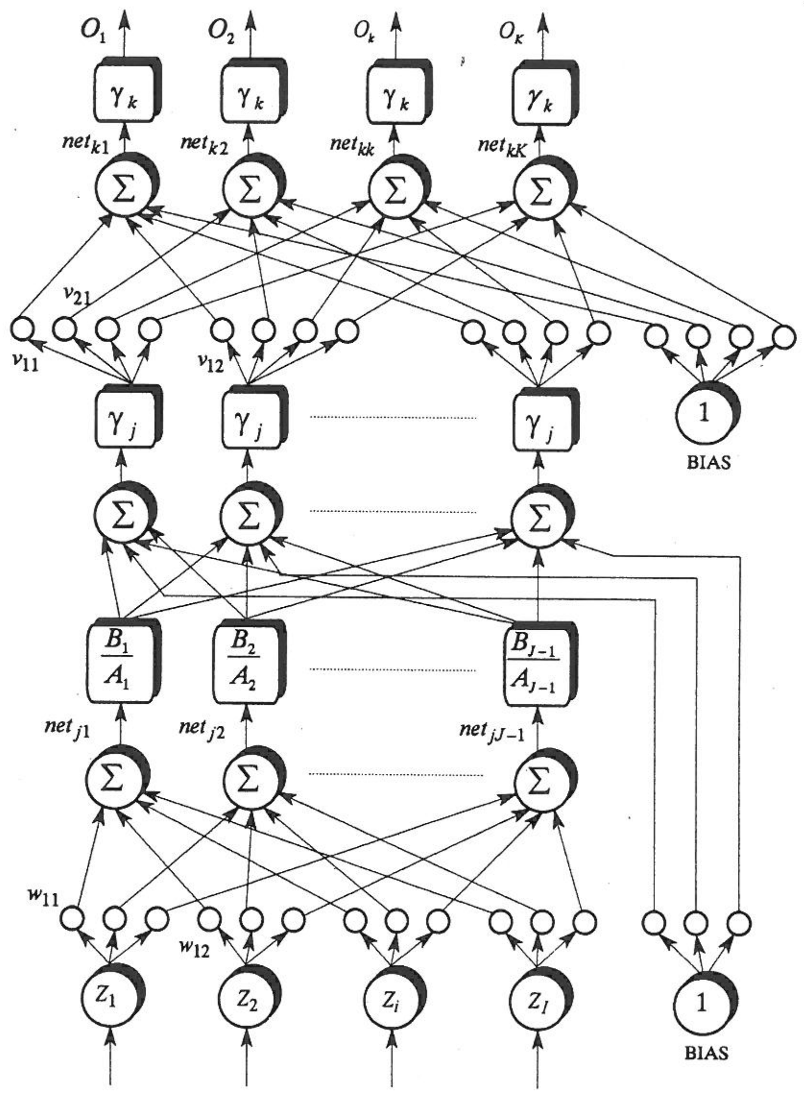

Fig. 2. Dynamic neural network (DNN)

where $k = 1, 2, \ldots, K$ is the number of neural network outputs.

To determine the optimal network parameters that minimize the index performance E, a gradient method can be applied. Iteratively, the optimal parameters (both the weights and filter coefficients) are approximated by moving in the direction of the steepest descent [Zu92]:

$$\vartheta_{new} = \vartheta_{old} + \Delta\vartheta \tag{7}$$

$$\Delta\vartheta = -\eta\nabla E = -\eta\frac{\partial E}{\partial\vartheta} \tag{8}$$

where η is a user-selected positive learning constant (learning rate). The choice of the learning constant depends strongly on the class of the learning problem and on the network architecture. The learning rate values ranging from 10^{-3} to 10 have been reported throughout the technical literature as successful for many computational back-propagation experiments. For large constants, the learning speed can be drastically increased ; however, the learning may not be exact, with tendencies to overshoot, or it may never be stabilized at any minimum.

To accelerate the convergence of the error-back propagation learning algorithm given in (7), the momentum method can be applied. The method [Zu92] is given in (9) and involves supplementing the current learning parameter adjustment (8) with a fraction of the most recent parameter adjustment. This is usually done according to the formula

$$\begin{aligned}\Delta\vartheta(n) &= -\eta E\nabla E(n) + \alpha\Delta\vartheta(n-1)\\ &= -\eta\frac{\partial E(n)}{\partial\vartheta(n)} + \alpha\Delta\vartheta(n-1) \qquad (9)\end{aligned}$$

where the arguments n and $n-1$ are used to indicate the current and the most recent training step (instant time), respectively, and a is a user-selected positive momentum constant. Typically, α is chosen between 0.1 and 0.8. The second term in (9) indicating a scaled most recent adjustment of parameter ϑ, is called the momentum term.

To simplify the derivation of the learning algorithm, a linear time shifting operator can be defined by equation (10).

$$\begin{aligned}&[\widetilde{y}(n)] = \frac{B(z)}{A(z)}[\mathrm{net}(n)]\\ &z^{-i}[\mathrm{net}(n)] = \mathrm{net}(n-i) \qquad (10)\\ &A(z)[\widetilde{y}(z)] = \widetilde{y}(n) + a_1\widetilde{y}(n-1) + a_2\widetilde{y}(n-2)\\ &B(z)[\mathrm{net}(n)] = b_0\,\mathrm{net}(n) + b_1\,\mathrm{net}(n-1)\\ &\qquad\qquad + b_2\,\mathrm{net}(n-2)\end{aligned}$$

According to the Fig. 1 it is obvious that:

$$\begin{aligned}\frac{\partial y(n)}{\partial\vartheta(n)} &= \frac{\partial y(n)}{\partial\overline{y}(n)}\frac{\partial\overline{y}(n)}{\partial\widetilde{y}(n)}\frac{\partial\widetilde{y}(n)}{\partial\vartheta(n)}\\ &= \gamma'\frac{\partial\overline{y}(n)}{\partial\widetilde{y}(n)}\frac{\partial\widetilde{y}(n)}{\partial\vartheta(n)} \qquad (11)\end{aligned}$$

Therefore, the used activation function has to be differentiable.

Using the time shifting operator defined in (10), four cases can be distinguished:

1) ϑ is a filter coefficient of the numerator $B(z)$:

$$\left.\frac{\partial[\widetilde{y}(n)]}{\partial\vartheta}\right|_{\vartheta=b_i} = [D_\vartheta(n)] = \frac{z^{-1}}{A(z)}[\mathrm{net}(n)] \qquad (12)$$

2) ϑ is a filter coefficient of the denominator $A(z)$:

$$\left.\frac{\partial[\widetilde{y}(n)]}{\partial\vartheta}\right|_{\vartheta=a_i} = [D_\vartheta(n)] = \frac{-z^{-1}}{A(z)}[\widetilde{y}(n)] \qquad (13)$$

3) ϑ is a neuron input weight:

$$\left.\frac{\partial[\widetilde{y}(n)]}{\partial\vartheta}\right|_{\vartheta=w_j} = [D_\vartheta(n)] = \frac{B(z)}{A(z)}[u_j(n)] \qquad (14)$$

4) ϑ is a neuron threshold:

$$\left.\frac{\partial y(n)}{\partial\vartheta}\right|_{\vartheta=w_y} = \frac{\partial\gamma}{\partial w_y} \qquad (15)$$

$D_\vartheta(n)$ is a current parameter state within the dynamic filters described on the right side of equations (12), (13), and (14).

Thus, to determine the change of the dynamic neuron activity depending on a filter and weight parameters, the gradient has to be calculated through time by the memory of the used filter.

The weight adjustment in the output layer (Fig. 2) can be obtained by expansion (17).

$$\left.\frac{\partial E(n)}{\partial\vartheta(n)}\right|_{\vartheta=v_{kj}} = \frac{\partial E(n)}{\partial O_k(n)}\frac{\partial O_k(n)}{\partial\,\mathrm{net}_k(n)}\frac{\partial\,\mathrm{net}_k(n)}{\partial\vartheta(n)} \qquad (16)$$

$$\left.\frac{\partial E(n)}{\partial\vartheta(n)}\right|_{\vartheta=v_{kj}} = -\,(d_k(n) - O_k(n))y_j(n) \qquad (17)$$

Finally, a measure of performance must be specified. All error measures will be reported using non-dimensional error index NRMS, Normalized Root Mean Square error. "Normalized" means that the root mean square is divided by the standard deviation of the target data [La87]. Thus the resulting error index, or index of accuracy, is insensitive to the dynamic range of the learning data, and allows easy comparison with other learning algorithms, independent of machine architecture or simulator implementation.

The computer programs for proposed dynamic neural network were written in C++ Programming language and run on the Intel i860 RISC processor (33Mhz).

4. Prediction the Glass–Mackey Time Series

Many conventional signal processing tests, such as correlation function analysis, cannot distinguish deterministic chaotic behaviour from stochastic noise. Particularly difficult systems to predict are those that are nonlinear and chaotic. It is known that chaos has a technical definition based on nonlinear, dynamic systems theory

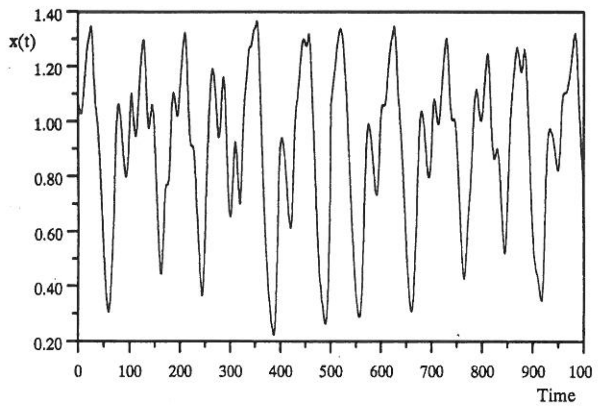

Fig. 3. The Glass–Mackey Time Series with $a = 0.1$, $b = 0.2$, $t = 30$

[La87]. Examples of chaotic systems in nature include chemical reactions, plasma physics, turbulence in fluids, lasers, to name a few. When parameters are varied, chaotic systems also display the full range of nonlinear behaviour (limit cycles, fixed points, etc.). Therefore chaotic systems provide a good testbed in which to investigate techniques of nonlinear signal processing, such as neural networks.

Lapedes and Farber [La87] suggested the Glass–Mackey time series as a good benchmark for learning algorithms, because it has a simple definition, yet its elements are hard to predict (the series is chaotic).

Glass–Mackey equation given in (18) is a nonlinear differential delay equation with an initial condition specified by an initial function defined over a strip with τ.

$$\dot{x} = \frac{ax(t-\tau)}{1+x^{10}(t-\tau)} - bx(t) \qquad (18)$$

Choosing the initial function to be constant function, with $a = 0.2$, $b = 0.1$ and $\tau = 17$ yields a time series $x(t)$ obtained by equation (18), that is chaotic with a fractal attractor of dimension 2.1. Increasing τ to 30 yields more complicated evolution and fractal dimension (d_A) of 3.5. The time series for 1000 time steps for $\tau = 30$ (time in units of τ) is plotted in Fig. 3.

The goal of the task is to use known values of the time series up to the point $x(t)$, to predict the value $x(t+P)$ at some point P in the future. The standard method for this type of prediction is to create a mapping $f\,()$ as follows:

$$x(t+P) = f\,(x(t), x(t-\Delta), x(t-2\Delta), \ldots, x(t-m\Delta)) \qquad (19)$$

where P is a prediction time into the future, Δ is a time delay , and m is an integer.

According to the equation (19) an attractor can be reconstructed from a time series by using a set of time delayed samples of a series. By choosing $P = \Delta$ [La87] it is possible to predict the value of time series at any multiple of Δ time steps in the future, by feeding the output back into the input and iterating the solution. In this study we choose to use $P = \Delta = 6$, since results can be compared with previous experiments where $P = 6$. Takens theorem [Ta81] states the range for dimension of the attractor (d_A):

$$d_A < m+1 < 2d_A + 1 \qquad (20)$$

For τ we choose $m = 4$.

It is obvious that for $P = \Delta = 6$ and $m = 4$ the expansion (19) has the following form:

$$x(t+6) = f\,(x(t), x(t-6), x(t-12), x(t-18), x(t-24)) \qquad (21)$$

Takens theorem unfortunately gives no information on the form of the $f\,()$ in Eqn. (21). Therefore, it is necessary to point out that the neural networks provide a robust approximating procedure for continuos $f\,()$.

The network which will be used to predict the chaotic system (21) is given in Fig. 2. According to the equation (21) the input layer consists of 5 neurons (input buffer), and the output layer consists of one static neuron with linear activation function. For hidden layer we suggested 10 dynamic neurons. Lapedes and Farber [La 87] for the same task used 20 hidden static neurons arranged in two hidden layer architecture. For training the neural network described above, we used first 500 values plotted in Fig. 3. Training started with random weights values between -1 and $+1$, while the filter coefficients a_1 and a_2 were initialized to zeros to support a stable learning procedure. The network was trained with $\eta = 0.01$ and $\alpha = 0.8$ until the error index NRMS dropped to 0.03.

The trained network was used to predict new sets of values $x(t)$ in the future. In all test results the NRMS was less than 0.04.

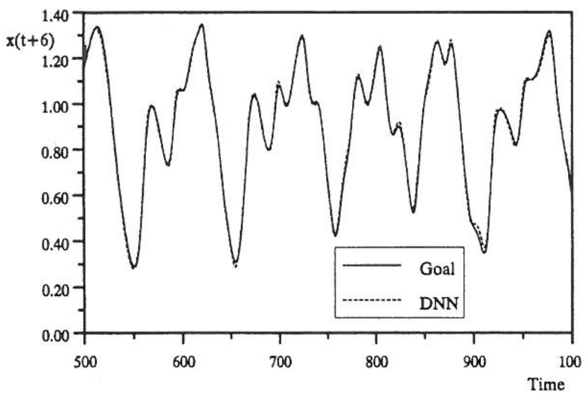

Fig. 4. Network evolution for test 1.

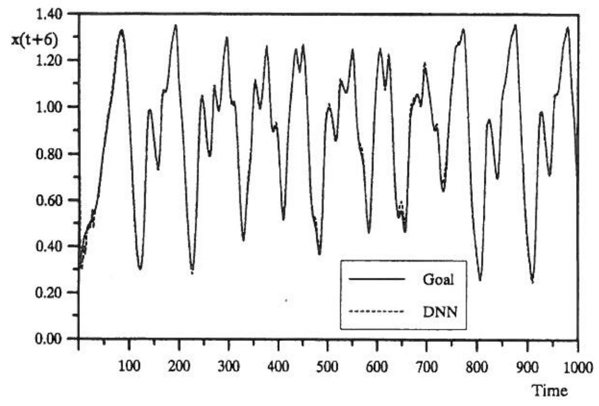

Fig. 5. Network evolution for test 2.

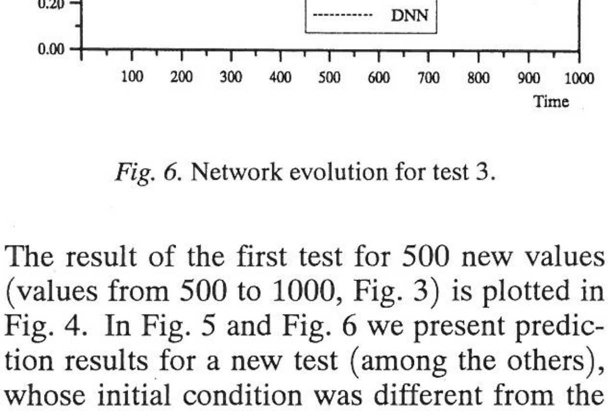

Fig. 6. Network evolution for test 3.

The result of the first test for 500 new values (values from 500 to 1000, Fig. 3) is plotted in Fig. 4. In Fig. 5 and Fig. 6 we present prediction results for a new test (among the others), whose initial condition was different from the one in the previous test. It is significant that test results are very similar for any other initial condition over a strip $\tau = 30$. Test results (Fig. 4–6) show that the proposed dynamic neural network can be successfully used in signal processing for prediction time series such as is nonlinear chaotic Glass–Mackey equation. It is interesting to note that the prediction of the chaotic time series done in this paper required the neural network of smaller size (less hidden layers and number of parameters) than in the benchmark paper from Lapedes and Farber [La87].

5. Identification of Nonlinear Dynamic System

As an even more interesting application of the proposed neural network algorithm based on dynamic elementary processor, the identification of the dynamic discrete-time nonlinear system is performed. The system behaviour is governed by the 1st order difference equation

$$x(n+1) = (0.9 - 0.003x(n))x(n) + 0.2u(n) \tag{22}$$

with sampling time $T_0 = 1$ s and a state-dependent time constant of about $T \approx 10$ s.

Such system is difficult to identify by classical methods when the mathematical structure of nonlinearity is unknown, because nonlinearity cannot be separated from the linear dynamics like e.g. a Hammerstein model. In order to obtain a good model of a nonlinear process, it is important that the learning data cover the whole relevant state space and contain a rich spectrum of frequencies. Thus, the process is excited with a pseudo-random binary noise (PRBS) signal with amplitude modulation. Measurement data are spoiled with pink noise of variance $0.05x_{\max}$. The set of 621 data samples is plotted in Fig. 7.

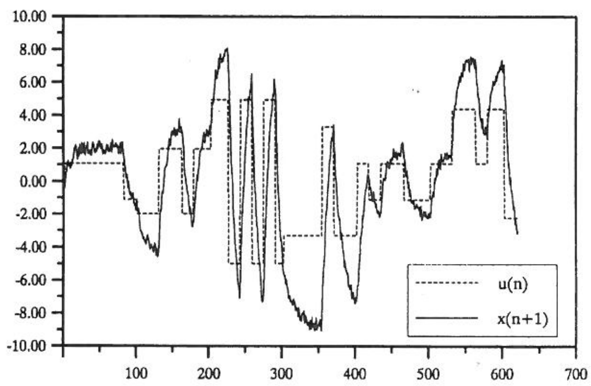

Fig. 7. The set of 621 data samples

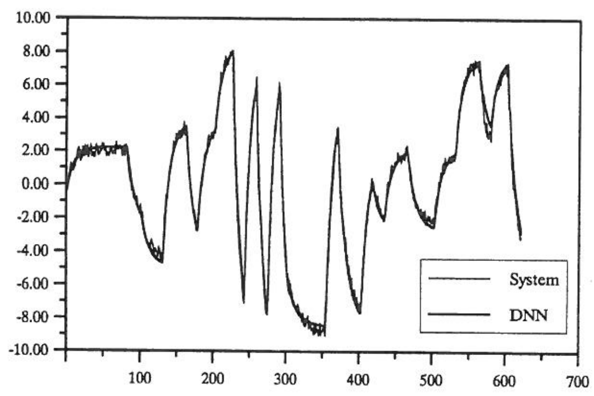

Fig. 8. Test result for 621 data samples

The identification of above described system is performed with neural network configuration as follows:

– input layer — 1 neuron $(u(n))$,

– hidden layer — 5 dynamic neurons,

– output layer — 1 static neuron $(x(n+1))$.

For training procedure the first 350 values of single input-output data set plotted in Fig. 7 are used. The network was trained with $\eta = 0.005$ and $\alpha = 0.8$ until the error index NRMS dropped to 0.05. Training started with random weights values between -1 and $+1$, while the filter coefficients a_1 and a_2 were initialized to zeros to support a stable learning procedure. In the recall or testing procedure the whole set of 621 data samples was used. Test results are plotted in Fig. 8. It is important to note that a given task was successfully performed with only one input neuron (ie., $x(n+1)$). This means, that one does not have to estimate the order of the identificied system in advance. In this way it is possible to avoid the modeling error due to the badly estimated system order.

6. Conclusion

Within this approach a Multi Layer Perceptron with distributed dynamics based on the DEP neuron model was proposed to predict a time series of nonlinear chaotic system, and for identification of a dynamic discrete-time nonlinear systems. An attempt was made within this approach to establish a basic dynamic neuron model, which processes multi inputs and does not require past values of the process measurements or prior information about its activity functions. The main advantage of proposed dynamic neuron model is that it reduces the network input space, and offers a great potential in solving many problems that occur in system modelling, with a special emphasis on the systems with characteristics such as nonlinearity, time delays, saturation or time-varying parameters. Therefore, in future work some efforts in this direction will be made, and the author hopes that then will be presented in some further paper.

7. References

[Ay94] M. AYOUBI, Nonlinear Dynamic Systems Identification with Dynamic Neural Networks for Fault Diagnosis in Technical Processes, Proceedings of IIIE, pp. 2120–2125, 1994

[Ch93] A. G. CHASSIAKOS, E. B. KOSMATOPOULOS and M. A. CHRISTODOULOU, Modeling of Robot Dynamics by Neural Networks with Dynamic Neurons, Neural Networks in Robotics, A.G. Bekey, K.Y. Goldberg (Eds.) Kluwe Acad. Publ., Boston, MA, pp. 165–176, 1993

[Cy89] G. CYBENKO, Approximations by Superposition of a Sigmoidal Function, Mathematics of Control, Signals, and Systems, Vol. 2, pp.303–314, 1989

[Du93] D. MAJETIĆ and V. KECMAN, Approximation Capabilities of ANNs as Function of the Number of Hidden Layer Neurons and the Type of Activation Function, Technical report TR93–YUSA–01, Massachusetts Institute of Technology, Cambridge, USA, pp. 31–50, 1993

[Fu89] K. FUNAHASHI, On The Approximate Realization of Continuous Mappings by Neural Networks, Neural Networks, vol. 2, pp. 183–192, 1989

[Gu92] M. M GUPTA, D. H. RAO, and J. GAO, Learning and Adaptation in Neural Control of High-Order Linear Systems, Proceedings of American Control Conference, ACC 92/FM2, Chicago, IL., pp. 3044–3048, 1992

[Gu93] M. M. GUPTA, D. H. RAO and P. N. NIKIFORUK, Dynamic Neural Network Based Inverse Kinematic Transformation, IFAC 93, Vol. 3, pp. 289–296, 1993

[Ko91] E. B. KOSMATOPOULOS, A. K. CHASSIAKOS, and M. A. CHRISTODOULOU, Robot Identification Using Dynamical Neural Networks, Proceedings of 30th Conference on Decision and Control, IEEE Transaction on Neural Networks, Brighton, England, pp. 2934–2935, 1991

[Ko92] E. B. KOSMATOPOULOS, P. A. IOANNOU and M. A. CHRISTODOULOU, Identification of Nonlinear Systems Using New Dynamic Neural Network Structure, Proceedings of the 31st Conference on Decision and Control, Tucson, Arizona, pp. 20–25, 1992

[La87] A. LAPEDES and R. FARBER, Nonlinear Signal Processing Using Neural Networks: Prediction And System Modeling, Technical Report, Los Alamos National Laboratory, Los Alamos, New Mexico, 1987

[Na90] K. S. NARENDRA and K. PARTHASARATHY, Identification and Control od Dynamical Systems Using Neural Networks, IEEE Transactions on Neural Networks, Vol. 1, No. 1, pp. 4–27, 1990

[Ni90] R. H. NIELSEN, Neurocomputing, Addison–Wesly Publishing Company, 1990

[Ra93] D. H. RAO and M. M. GUPTA, A Multi-Functional Dynamic Neural Processor for Control Applications, Proceedings of American Control Conference, San Francisco, California, pp. 2902–2906, 1993

[Ta81] F. TAKENS, Detecting Strange Attractor In Turbulence, Lecture Notes in Mathematics, D. Rand, L. Young (editors), Springer Berlin, pp. 366, 1981

[Zu92] J. M. ZURADA, Artificial Neural Systems, W. P. Company, USA, 1992

Received: June, 1995
Accepted: September, 1995

Contact address:
Dubravko Majetić
Department of Control Engineering
Faculty of Mechanical Engineering
and Naval Architecture
University of Zagreb
Ivana Lučića 5
Zagreb, Croatia
tel: +385 1 611 944/348
fax: +385 1 514 535
telex: 22648 fsb CROATIA
e-mail: dubravko.majetic@x400.srce.hr

DUBRAVKO MAJETIĆ (1962) received the B.Sc. and M.Sc. degrees in Mechanical Engineering from the Faculty of Mechanical Engineering and Naval Architecture (FSB), University of Zagreb in 1988 and 1992. He is lecturer in Department of Control Engineering, FSB, University of Zagreb. Currently he is working toward Ph.D. degree in the field of Neural Networks. His interests include Artificial Intelligence, Neural Networks, Robotics, Automatic Control.