FACTA UNIVERSITATIS
Series: **Automatic Control and Robotics** Vol. 14, N^{o} 3, 2015, pp. 173 - 182

DATA DENOISING PROCEDURE FOR NEURAL NETWORK PERFORMANCE IMPROVEMENT

UDC (004.32.26:681.513.5)

Miroslav B. Milovanović, Dragan S. Antić, Saša S. Nikolić, Miodrag D. Spasić, Staniša Lj. Perić, Marko T. Milojković

University of Niš, Faculty of Electronic Engineering, Department of Control Systems, Niš, Republic of Serbia

Abstract. *This paper will present training data denoising procedure for neural network performance improvement. Performance improvement will be measured by evaluation criterion which is based on a training estimation error and signal strength factor. Strength factor will be obtained by applying denoising method on a default training signal. The method is based on a noise removal procedure performed on the original signal in a manner which is defined by the proposed algorithm. Ten different processed signals are obtained from the performed method on a default noisy signal. Those signals are then used as a training data for the nonlinear autoregressive neural network learning phase. Empirical comparisons are made at the end, and they show that the proposed denoising procedure is an effective way to improve network performances when the training set possesses the significant noise component.*

Key words: *recurrent neural network, training data, noise removal, linear regression, optimization procedure*

1. INTRODUCTION

This paper presents a procedure for improving estimation performances of recurrent neural networks. The performance improvement will be achieved by performing the training data denoising procedure. Complex and nonlinear training database will be considered as a suitable test case for the proposed procedure. A type of neural network which will be used in the paper is determined according to the previous artificial usage in the field of noise reduction. Neural network types which are good in data estimations are

Received November 9, 2015
Corresponding author: Miroslav B. Milovanović
University of Niš, Faculty of Electronic Engineering, Department of Control Systems, Aleksandra Medvedeva 14, 18000 Niš, Republic of Serbia
E-mail: miroslav.b.milovanovic@elfak.ni.ac.rs

presented in [1]. Backpropagation learning algorithm [BP] is one of widely used algorithms for the estimation purposes and analysis of nonlinear data [2-8]. BP basic feature is that it is based on the steepest gradient descent method [9] and can be successfully used for optimization and approximation purposes. The lack of BP algorithm may be a risk of trapping in a local minimum, in which case specific mathematical apparatuses should be performed. Backpropagation learning algorithms give optimal network performances when they are applied to multilayer feedforward neural networks. Networks are expected to learn noisy data, to generalize the model well and to estimate specified output values with the desired accuracy. Recurrent and feedforward multilayer neural networks are also used to solve estimation and noise reduction problems. Recurrent neural networks are used in [10] to preprocess data and decrease the effects of noise on the output results. In [11] and [12] are presented neural networks with additive noise in the desired signal and radial basis function network respectively, for solving different problems of noisy signals. In [13] is developed a thresholding neural network for adaptive noise reduction. Recurrent neural network, which is used for noisy time series predictions, is presented in [14]. Predictions of noisy time series data are also commonly done using various statistical models. Different recurrent neural networks whose purpose is to predict financial time series with disturbances are presented in [15].

The focus of our paper is in proposing the training data optimization method. The method will be efficient for noisy nonlinear signals which should be precisely estimated by the neural network. It is shown in many practical examples [16-20] that preprocessing training data could result with an error minimization of a network training process. For this purpose, a method for removing noise data from the training signal will be presented in the next sections. The main goal is to find an appropriate compromise between the volume of removed data, network performances and reduction of training computation time.

2. ERROR COMPONENTS DEFINITION AND LINEAR REGRESSION METHOD

Expanded bias-variance decomposition is used as an initial analysis in the process of improving the performance of the network in [21]. Basically, this paper assumes that the estimation error depends on two components: bias and variance. It is considered that a poorly chosen network model affects the appearance of a large bias, while untrained new network can be affected with a large variance. Variance error usually occurs when a neural network becomes loaded with a large number of parameters which is necessary to estimate. Squared bias for specific input values x can be defined as:

$$B_{sq} = (E_D[f(x;D)] - E[y|x])^2, \tag{1}$$

where E_D is the expectation operator which defines mean convergence, D is the training set defined as: $D = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\} = \{D^1, D^2, ..., D^m\}$, $f(x;D)$ is the regression estimator for a training set, and $E[y|x]$ is desired constant value (a range). If the goal is to minimize the mean squared error, the best results could be obtained by forming relationships between x and $E[y|x]$. The squared variance definition can be represented as:

$$V_{sq} = E_D\left[(f(x;D) - E_D[f(x;D)])^2\right]. \tag{2}$$

Overall values of bias errors and variance errors could be found from dependences on the number of training sets *m*:

$$B_{terr}(x) = \sum_{k=1}^{m} \left[f(x;D^k) - E\left[y|x \right] \right]^2 , \tag{3}$$

$$V_{terr}(x) = \sum_{k=1}^{m} \left[f(x;D^k) - f(x) \right]^2 , \tag{4}$$

where $B_{terr}(x)$ is total bias error and $V_{terr}(x)$ is total variance error. Bias and variance are two elemental parts of total error which exist during the prediction procedure. Third part of the total error, which represents the mean - squared error of the sum and variance, is introduced in [19]:

$$E_{mv} = \frac{1}{m} \sum_{k=1}^{m} (f(x;D^k) - E\ [y|x])^2 , \tag{5}$$

where E_{mv} is the mean-squared error. Total error in [21] is denoted as expected loss and it is composed of three parts: bias, variance and noise. It can be observed in comparison to [21] that the mean-squared error part is replaced with noise parameter.Due to that, general error can be represented as:

$$G_{err} = c_1 \cdot N_t(x) + B_{terr}(x) + c_2 \cdot V_{terr}(x) , \tag{6}$$

where $N_t(x)$ is noise parameter. The noise loss is, in general, small and it is equal to the difference between prediction values and target values. Coefficients c_1 and c_2 could be chosen with different values for different error functions, and they are highly adjustable. Procedure of neural network performance improvement, by reducing noise parameter error from the equation (6), will be the main goal in our paper.

Randomly generated noisy signal will undergo linear regression procedure in the next section with the purpose of $N_t(x)$ minimization. The basic principles of linear regression are presented in [22]. The linearity is provided with the formula:

$$y_i = x_{i1}\beta_1 + x_{i2}\beta_2 + ... + x_{ik}\beta_k + \varepsilon_i , \tag{7}$$

where is presented the model that defines the linear relationship between *y* and $x_1,x_2,...,x_k$. The parameter ε_i represents disturbance in the *i*-th observation. Noisy training dataset which will be used for a case study is presented in the next section. The linear regression method will be used to find a straight line modeling of a two-dimensional training dataset in our paper. For this purpose will be used equation of the straight line following the example from the equation (7):

$$y = \alpha x + \beta , \tag{8}$$

where α and β are parameters which could be calculated for known values of *x* and *y*. Mean values of *x* and *y* could be found from:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \ \overline{y} = \frac{\sum_{i=1}^{n} y_i}{n}, \tag{9}$$

where n is the number of training dataset elements. Parameters α and β could be easily found from (9):

$$\alpha = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$
$$\beta = \bar{y} - \alpha\bar{x} \tag{10}$$

The linear regression procedure for the generated training dataset will be performed in the following section. Also, an algorithm for training data optimization and noise removal process will be proposed.

3. TRAINING DATASET OPTIMIZATION PROCEDURE

Neural network learning data is obtained from Matlab software package. The training data set is formed by generating random noisy input signal using the specified options:

- Signal amplitude: 0.1
- Number of samples: 1000
- Time period: 10 sec

Preview of generated signal is presented in Fig 1. The main task of neural network will be to estimate the proposed input signal with the smallest error possible to achieve. Noise removal procedure will be performed in order to minimize the error approximations. In that manner, preprocessed signal will be simplified for the neural network estimation procedure. It should be noted that the focus of this paper is on the presentation of compromises that need to be made between the level of denoising procedure and the quality of approximation: as much noise is removed from the input signal, the quality of the signal will be poorer for information which are lost. On the other hand, a simplified signal will increase neural network estimation speed and reduce its overall training error. In the sequel of this paper empirical analysis of this compromise for different degrees of denoising process will be presented.

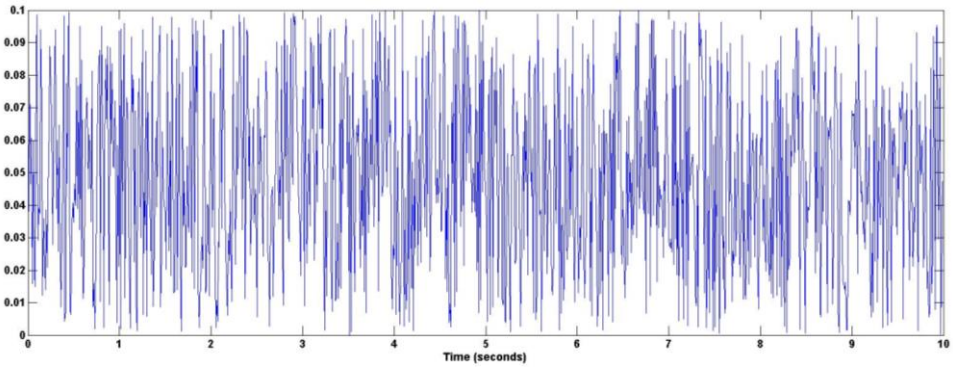

Fig. 1 Generated random training signal

Linear regression equation, which presents a starting point for signal modification, is obtained by applying (9) and (10) on the signal from Fig.1:

$$y = -0.00041x + 0.051. \tag{11}$$

Linear regression (LR) straight line, which is presented in Fig. 2, is drawn on the basis of the equation (11) and inserted across noisy signal graph. Every noisy two dimensional signal could be approximated by an LR straight line modelling. The problem, for which we will present solution in this paper, is how to select a proper quantity of data which should be removed from the default signal regarding the LR line. Algorithm 1 presents the procedure for signal noise removal and a proper way for obtaining different training data for the purpose of neural network learning phase.

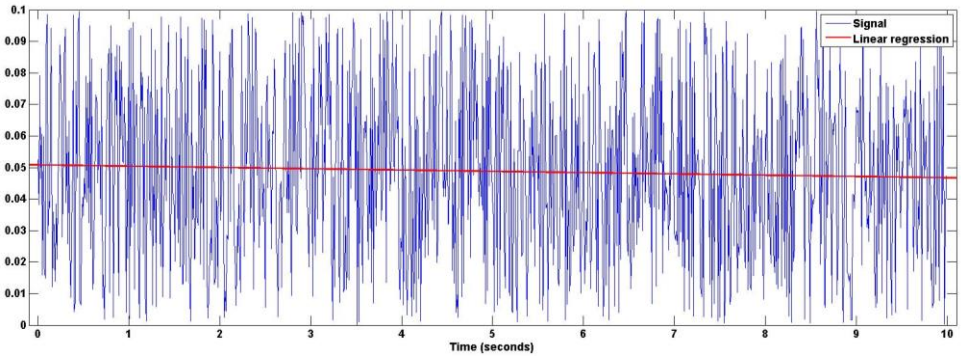

Fig. 2 Default signal and calculated LR straight line

Algorithm 1 Construction procedure of denoised signals

1. Input vector y – default signal values on y-axis (y axis on Fig.1).
2. Computations:
   1) Vector D – distances between each signal point and linear regression line
   2) Dmax – position of maximal distance point (between signal and linear regression line)
   3) Dmin – position of minimal distance point (between signal and linear regression line)
   4) NOISEmax – distance between Dmax and corresponding point on the linear regression line
   5) NOISEmin – distance between Dmin and corresponding point on the linear regression line
   6) Tmax – point on the linear regression line which corresponds to Dmax
   7) Tmin – point on the linear regression line which corresponds to Dmin
3. Determine 10 different values for parameter Gama which control what percent of noise will be removed from the signal (in the range 0 to 100%, step size is 10%).
4. For each value of parameter Gama find denoised signal in the form of a vector.
5. Group the vectors from step 4 into the matrix. Matrix dimensions are 10x1000.
6. Use Matrix to plot denoised signals. Matrix data present neural network learning dataset with 10 processed signals.

Procedure from algorithm 1 is applied to default signal from Fig. 1. The original signal and 10 modified signals obtained from the different quantities of the removed noise (from default signal) are presented in Fig. 3. The original signal is labeled with y. Ten obtained signals with 100% to 10% of removed noise are labeled as $d_{ns1}, d_{ns2}, d_{ns3}, ..., d_{ns10}$, respectively. In the next section, neural network structure will be proposed, and training procedures for obtained signals from Fig. 3 will be presented.

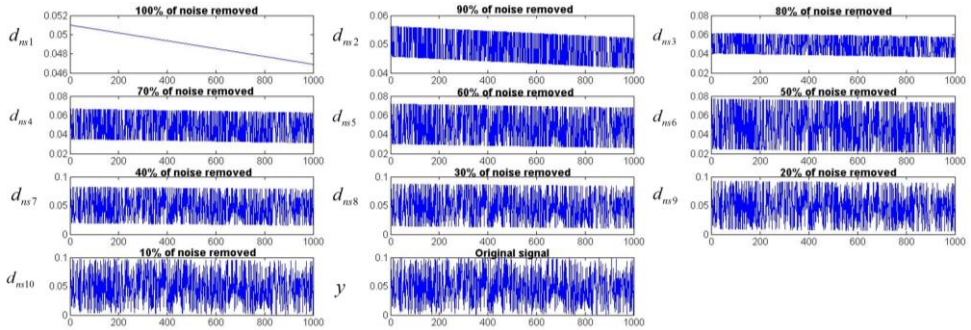

Fig. 3 Original and denoised signals

4. NEURAL NETWORK STRUCTURE AND TRAINING PROCEDURES

Neural network which will be chosen for training and testing procedures is determined by good approximation abilities. Recurrent neural network (RNN) is selected for experimental purposes in this paper because it is commonly used in the field of processing noisy signals. Recurrent neural networks are dynamic networks that have recurrent (feedback) connections. They possess memory and depend on input sequence history and current input values. Dynamic networks are suited for filtering purposes because they can be trained to learn time-varying patterns. They have slower response time compared to feedforward networks, which is their main disadvantage. Elemental recurrent neural network structures and training possibilities are presented in [23] with the accent on the learning rate parameter. The learning rate, which represents an important training parameter, can be explained as a constant which determines the degree of weight coefficient changes during learning procedure. A neural network with one hidden layer and sufficient number of neurons in the hidden layer is capable of approximating almost any continuous function. Neural network with a single hidden layer is selected because of that. Another reason for selecting one hidden layer is the desire to avoid more complex network structures (potential danger of overfitting and extra computation time). The hidden layer should possess one half to three times more neurons than the input layer, and this factor depends on the complexity of the problem. It is advisable to double the number of hidden neurons until the satisfactory network performances are obtained. Twelve neurons in the hidden layer are finally chosen, based on performed empirical tests. The neural network transfer function is selected to be sigmoid function. This function is used for time series data because it is nonlinear and continuously differentiable. Those characteristics are good enough reasons to implement sigmoid activation function inside the network. Specific type of the network which will be used for testing

procedures will be simple nonlinear autoregressive recurrent neural network (NARX). Time series NARX is described in details in [24], [25]. Training type which will be performed is mathematically straightforward and recurrent. Matlab function which will accomplish this type of training is called *trainbr*, which is applicable for noisy datasets in conjunction with the Bayesian regularization training function.

The strength factor of performed denoised procedures will be proposed after we specified neural network structure which will be used. This factor presents an important measure of signal quality after simplification procedure is performed. Strength factor S_f is determined by comparing specific denoised signal and the original signal by follows:

$$S_f = \frac{\sum_{m=1}^{1000} \left| d_{ns}(m) - l_{rg}(m) \right|}{\sum_{m=1}^{1000} \left| y(m) - l_{rg}(m) \right|}, \tag{12}$$

where d_{ns} is the specific denoised signal from Fig. 3, y is the default signal from Fig. 1, and l_{rg} is the LR straight line value determined by (11). Relations are based on signal range comparisons where referent positions are on l_{rg} line. Values for each strength factor are presented in Table 1.

Table 1 Strength factor values

Signal	d_{ns1}	d_{ns2}	d_{ns3}	d_{ns4}	d_{ns5}	d_{ns6}	d_{ns7}	d_{ns8}	d_{ns9}	d_{ns10}
Strength factor	S_{f1}	S_{f2}	S_{f3}	S_{f4}	S_{f5}	S_{f6}	S_{f7}	S_{f8}	S_{f9}	S_{f10}
Values	0	0,169	0,320	0,456	0,576	0,681	0,772	0,849	0,912	0,964

Neural network training procedures are performed using Matlab software package on original and denoised signals. Training results are presented in Table 2. The mean squared estimation error is labeled as e_{rr}, training speed as t_{sp}, strength factor is already marked as S_f, and the most important parameter for performance evaluation is labeled with γ.

Table 2 Training results and evaluation criterion values

Signal	Estimation error (e_{rr})	Iterations	Training speed (t_{sp})	Strength factor (S_f)	Evaluation criterion (γ)
d_{ns1}	$3{,}81 \cdot 10^{-10}$	6	1,01 s	0	0
d_{ns2}	$2{,}43 \cdot 10^{-5}$	15	0,90 s	0,169	5,7
d_{ns3}	$8{,}73 \cdot 10^{-5}$	14	0,93 s	0,320	11,37
d_{ns4}	$1{,}8 \cdot 10^{-4}$	9	0,96 s	0,456	16,15
d_{ns5}	$2{,}92 \cdot 10^{-4}$	9	0,95	0,576	19,53
d_{ns6}	$4{,}09 \cdot 10^{-4}$	11	0,98	0,681	23,2
d_{ns7}	$5{,}46 \cdot 10^{-4}$	10	0,90	0,772	25,91
d_{ns8}	$6{,}45 \cdot 10^{-4}$	12	0,91	0,849	28,8
d_{ns9}	$7{,}17 \cdot 10^{-4}$	10	0,87	0,912	31,045
d_{ns10}	$8{,}19 \cdot 10^{-4}$	9	0,86	0,964	32,48
y	$1{,}17 \cdot 10^{-3}$	9	1,21	1	29,23

It could be easily concluded from the table that the number of iterations and training speed, which are the required parameters for training completion, are not changing significantly for different processed signals. Because of that, they will not be important for evaluation criterion forming procedure. Further, it is obvious that signal estimation errors are increasing with the increase of strength factor. The best case would be to have the largest S_f in combination with the smallest e_{rr} possible. In accordance with that, evaluation criterion γ is calculated in the table as:

$$\gamma = \frac{S_f^{\,2}}{\sqrt{e_{rr}}}. \tag{13}$$

Evaluation criterion is formed in a manner that favors signal strength and decreases error factor. It can be concluded from evaluation criterion values presented in Table 2 that the best network performances are obtained for d_{ns10}. This denoised signal is formed with 10% of removed noise from the original signal. Overall network performances according to evaluation criterion are better from performances obtained using the original signal y. The signal d_{ns9} is also suitable for usage, whereas network performances after using this signal are still better from performances obtained by processing original signal y. General conclusion is that the proposed data denoising procedure could improve network performances when it is performed on noisy training signals. The method is most adequate when it is used for noise removal between 10% - 20% from the original signal. In those cases, the method provides optimized training procedures of recurrent neural networks. The proposed method will be tested on other types of training data and different neural network types in future researches, with the purpose to form a universal preprocessing method.

5. CONCLUSION

In this paper we presented the method for improving neural network performances by preprocessing training data. The method is formed by defining all components of error signal, their thorough examination and focusing on the noise component. A random training signal which possesses noisy characteristic is formed in the Matlab software package. First, a linear regression procedure is performed on this noisy signal with the purpose to obtain a straight line modeling of a two-dimensional data set. Denoising algorithm is then presented in detail and used for processing the signal. Ten different signals which are obtained by denoising procedure are then used as training data for nonlinear autoregressive neural network learning procedures. Evaluation criterion, which is designed to evaluate network performances, is based on the mean squared estimation error values and signal strength factor. Signal strength factor presents newly developed performance parameter based on signal quality. On the basis of evaluation criterion, it is concluded that neural network showed improved learning performances when 10% to 20% of noisy signal data is removed from the default signal. The method represents a basic platform for future researches on a topic of training data preprocessing methods which could improve general network abilities.

Acknowledgement: *This paper was realized as a part of the projects "Studying climate change and its influence on the environment: impacts, adaptation and mitigation" (III 43007), and "Research and development of new generation wind turbines of high-energy efficiency" (TR 35005), financed by the Ministry of Education, Science and Technological Development of the Republic of Serbia within the framework of integrated and interdisciplinary research for the period 2011-2016.*

REFERENCES

[1] F. Scarselli, A. C. Tsoi, "Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results," Neural Networks, vol. 11, no. 1, pp. 15–37, 1998. [Online]. Available: http://dx.doi.org/10.1016/S0893-6080(97)00097-X

[2] T. Jayalakshmi, A. Santhakumaran, "Statistical normalization and back propagation for classification," International Journal of Computer Theory and Engineering, vol. 3, no. 1, pp. 89–93, 2011. [Online]. Available: http://www.ijcte.org/papers/288-L052.pdf

[3] K. Shihab, "A backpropagation neural network for computer network security," Journal of Computer Science, vol. 2, no. 9, pp. 710–715, 2006. [Online]. Available: http://docsdrive.com/pdfs/sciencepublications/jcssp/2006/710-715.pdf

[4] S. Solanki, H. B. Jethva, "Modified back propagation algorithm of feed forward networks," International Journal of Innovative Technology and Exploring Engineering, vol. 2, no. 6, pp. 131–134,2013. [Online]. Available: http://ijitee.org/attachments/File/v2i6/F0816052613.pdf

[5] P. J. Werbos, "Backpropagation through time: What it does and how to do it," Proceedings of the IEEE, vol. 78, no. 10, pp. 1550–1560, 1990. [Online]. Available: http://dx.doi.org/10.1109/5.58337

[6] O. Al-Allaf, "Improving the performance of backpropagation neural network algorithm for image compression/decompression system," Journal of Computer Science, vol. 6, no. 11, pp. 1347–1354, 2010. [Online]. Available: http://94.142.32.98/conferences/ITC12/AllPapers/PA_13.pdf

[7] M.-B. Rădac, R.-E. Precup, E. M. Petriu, "Model-free primitive-based iterative learning control approach to trajectory tracking of MIMO systems with experimental validation," IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 11, pp. 2925–2938, 2015, [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2015.2460258

[8] M.-B. Rădac, R.-E. Precup, E. M. Petriu, St. Preitl, "Application of IFT and SPSA to servo system control," IEEE Transactions on Neural Networks, vol. 22, no. 12, part 2, pp. 2363–2375, 2011. [Online]. Available: http://dx.doi.org/10.1109/TNN.2011.2173804

[9] M. Z. Rehman, N. M. Nawi, "Improving the accuracy of gradient descent back propagation algorithm (GDAM) on classification problems," International Journal on New Computer Architectures and Their Applications, vol. 1, no. 4, pp. 838–847, 2011. [Online]. Available: http://sdiwc.net/digital-library/improving-the-accuracy-of-gradient-descent-back-propagationalgorithm-gdam-on-classification-problems.html

[10] J. R. Dorronsoro, V. López, C. S. Cruz, J. A. Sigüenza, "Autoassociative neural networks and noise filtering," IEEE Transactions on Signal Processing, vol. 51, no. 5, pp. 1431–1438, 2003. [Online]. Available: http://dx.doi.org/10.1109/TSP.2003.810276

[11] C. Wang, J. C. Principe, "Training neural networks with additive noise in the desired signal," IEEE Transactions on Neural Networks, vol. 10, no. 6, pp. 1511–1517, 1999. [Online]. Available: http://dx.doi.org/10.1109/72.809097

[12] D. G. Khairnar, S. N. Merchant, U. B. Desai, "Radar signal detection in non-Gaussian noise using RBF neural network," Journal of Computers, vol. 3, no. 1, pp. 32–39, 2008. [Online]. Available: http://www.ojs.academypublisher.com/index.php/jcp/article/view/03013239/366

[13] X.-P. Zhang, "Thresholding neural network for adaptive noise reduction," IEEE Transactions on Neural Networks, vol. 12, no. 3, pp. 567–584, 2001. [Online]. Available: http://dx.doi.org/10.1109/72.925559

[14] C. L. Giles, S. Lawrence, A. C. Tsoi, "Noisy time series prediction using a recurrent neural network and grammatical inference," Machine Learning, vol. 44, no. 1–2, pp. 161–183, 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1010884214864

[15] I. Kaastra, M. Boyd, "Designing a neural network for forecasting financial and economic time series," Neurocomputing, vol. 10, no. 3, pp. 215–236, 1996. [Online]. Available: http://dx.doi.org/10.1016/0925-2312(95)00039-9

[16] V. Nedeljković, M. Milosavljević, "On the influence of the training set data preprocessing on neural networks training," in *Proceedings of 11th IAPR International Conference on Pattern Recognition, Conference B: Pattern Recognition Methodology and Systems*, Hague, Netherlands, pp. 33–36, 1992. [Online]. Available: http://dx.doi.org/10.1109/ICPR.1992.201716

[17] O. E. de Noord, "The influence of data preprocessing on the robustness and parsimony of multivariate calibration models," Chemometrics and Intelligent Laboratory Systems, vol. 23, pp. 65–70, 1994. [Online]. Available: http://dx.doi.org/10.1016/0169-7439(93)E0065-C

[18] J. de Witt, "Adaptive filtering network for associative memory data preprocessing," in *Proceedings of the World Congress on Neural Networks*, San Diego, USA, vol. 4, pp. 34–38, 1994.

[19] H. H. Nguyen, C. W. Chan, "A comparison of data preprocessing strategies for neural network Modeling of oil production prediction," in *Proceedings of the Third IEEE International Conference on Cognitive Informatics*, Victoria, Canada, pp. 199–209, 2004. [Online]. Available: http://dx.doi.org/10.1109/COGINF.2004.1327476

[20] L.-O. Fedorovici, "A comparison between two character recognition approaches," Facta Universitatis Series: Automatic Control and Robotics, vol. 10, no. 2, pp. 125–140, 2011. [Online]. Available: http://facta.junis.ni.ac.rs/acar/acar201102/acar20110201.html

[21] S. Geman, E. Bienenstock, R. Doursat, "Neural networks and the bias/variance dilemma," Neural Computation, vol. 4 no. 1, pp. 1–58, 1992. [Online]. Available: http://dx.doi.org/10.1162/neco.1992.4.1.1

[22] L. Badri, M. Al-Azzo, "Modelling of wavelength detection of objects using Elman network modified covariance combination," The International Arab Journal of Information Technology, vol. 5, no. 3, pp. 265–272, 2008. [Online]. Available: http://ccis2k.org/iajit/PDF/vol.5,no.3/8-201.pdf

[23] H. Jaeger, "A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "Echo state network" approach," GMD Report 159, German National Research Center for Information Technology, pp. 1–48, 2002. [Online]. Available: http://www.pdx.edu/sites/www.pdx.edu.sysc/files/Jaeger_TrainingRNNsTutorial.2005.pdf

[24] D. Antić, M. Milovanović, S. Nikolić, M. Milojković, S. Perić, "Simulation model of magnetic levitation based on NARX neural networks," International Journal of Intelligent Systems and Applications, vol. 5, no. 5, pp. 25–32, 2013. [Online]. Available: http://dx.doi.org/10.5815/ijisa.2013.05.04

[25] D. Antić, M. Milovanović, S. Nikolić, S. Perić, M. Milojković, "Primena NARX neuronske mreže za simulaciju rada sistema magnetne levitacije," TEHNIKA, vol. 62, no. 3, pp. 473–479, 2013.