Dynamic Artificial Neural Network (DANN) MATLAB Toolbox for Time Series Analysis and Prediction

Khim Chhantyal Minh Hoang Håkon Viumdal Saba Mylvaganam

Faculty of Technology, Natural Sciences, and Maritime Sciences, University College of Southeast Norway,
{khim.chhantyal,hakon.viumdal,saba.mylvaganam}@usn.no, m.hoang1304@gmail.com

Abstract

MATLAB® Neural Network (NN) Toolbox can handle both static and dynamic neural networks. Using MATLAB® NN Toolbox with recurrent neural networks is not straight forward. We present a Dynamic Artificial Neural Network (DANN) MATLAB toolbox capable of handling fully connected neural networks for time-series analysis and predictions. Three different learning algorithms are incorporated in the MATLAB DANN toolbox: Back Propagation Through Time (BPTT) an offline learning algorithm and two online learning algorithms; Real Time Recurrent Learning (RTRL) and Extended Kalman Filter (EKF). In contrast to existing MATLAB® NN Toolbox, the presented MATLAB DANN toolbox has a possibility to perform the optimal tuning of network parameters using grid search method. Three different cases are used for testing three different learning algorithms. The simulation studies confirm that the developed MATLAB DANN toolbox can be easily used in time-series prediction applications successfully. Some of the essential features of the learning algorithms are seen in the graphical user interfaces discussed in the paper. In addition, installation guide for the MATLAB DANN toolbox is also given.

Keywords: dynamic artificial neural network (DANN), back propagation through time (BPTT), real-time recurrent learning (RTRL), extended Kalman filter (EKF), time series

1 Introduction

Artificial Neural Networks (ANN) are computational models consisting of many neurons in different layers with varying degrees of interconnections between them. The interconnection have weights assigned to them so that the ANNs can be tuned thus enabling them to learn and adapt. Feedforward or feedback networks are two broad classifications of ANNs. Feedforward ANNs use current inputs and current outputs, whereas, feedback ANNs use current and previous inputs and outputs. Feedback ANN performs time-series predictions and is a dynamic network. This type of network constitutes recurrent neural networks (RNN) either partially or fully connected depending on the extent of the feedback loops available in the network. Fully connected RNNs have interconnected feedback loops including self-feedback loops, whereas partially connected RNNs do not have self-feedback loops (Veelenturf and Gerez, 1999; Beale et al., 1992).

In an existing MATLAB® Neural Network Toolbox, there is a possibility to use feedforward ANN for static estimations and partially connected RNN for time-series predictions. This paper presents a MATLAB toolbox that can perform the empirical modeling using fully connected RNNs with three different learning algorithms. The following sections present the overview of the developed toolbox and the usage of the toolbox in three different practical applications.

2 Overview of Toolbox

The developed Dynamic Artificial Neural Network (DANN) toolbox consists of three main user interfaces, which are DANN Menu, Parameter Tuning, and Plot Menu. Each window consists of different elements as given below.

2.1 DANN Main

DANN Main is the main window of MATLAB DANN toolbox as shown in Figure 1. In this window, the user can upload the data set, divide data sets, select validation check, include bias, select learning algorithm, define learning parameters, select the number of previous inputs and outputs, and finally train the model.

2.1.1 Uploading data set

A user needs to upload his/her data set to train the model using MATLAB DANN toolbox. The format of the data set should be in '.mat' and each column should represent the variables in the model, where the last column is an output variable.

2.1.2 Division of data set

The uploaded data set should be divided into a training set, validation set, and testing set. A user can choose the percentage of data for training, validation and testing. Experience shows that 70% for the training set, 15% for both validation and testing works fine for any learning algorithms.

2.1.3 Validation check

A validation check is an option that prevents the over-fitting of the network. Over-fitting and under-fitting are most common problems encountered while dealing with

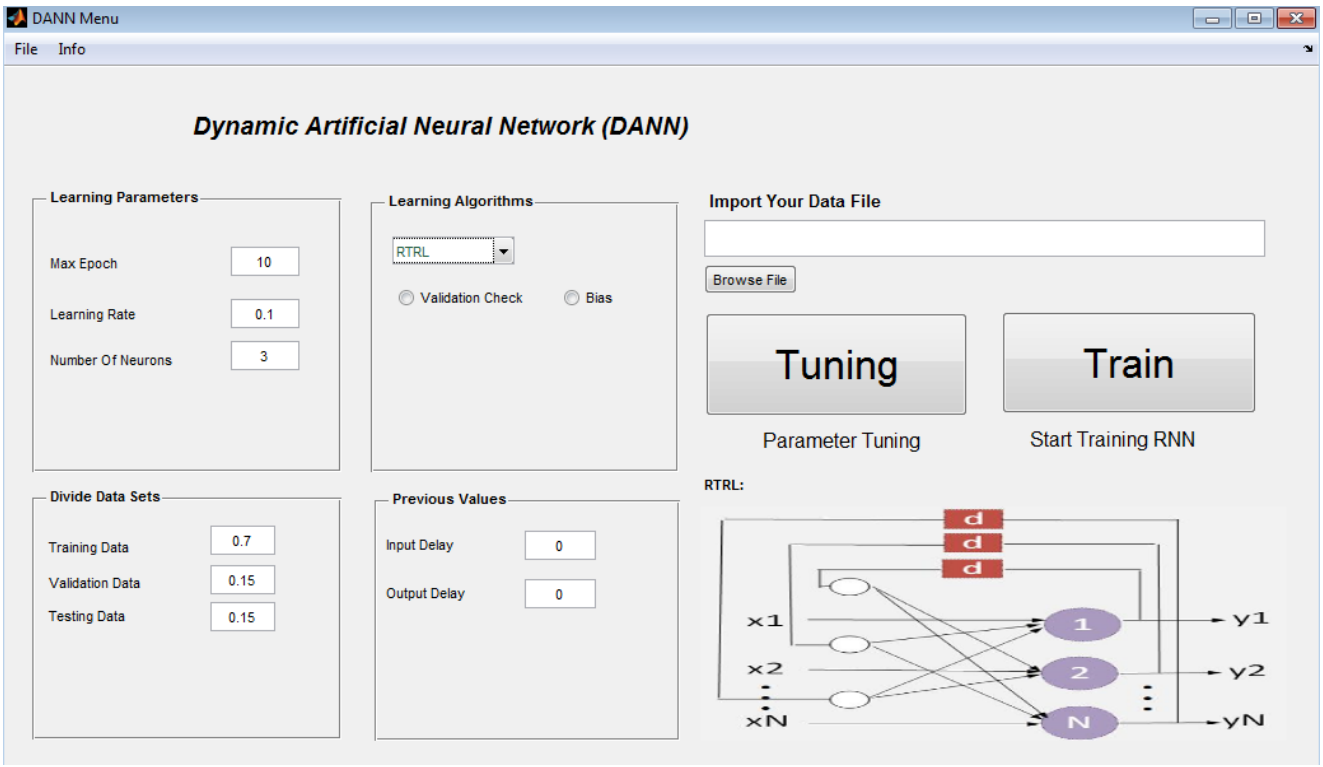

Figure 1. DANN Menu of MATLAB DANN toolbox: data upload, data division, selecting learning algorithms, and tuning learning parameters.

data models. Under-fitting can be improved by either tuning the learning rate or increasing the number of neurons in the network. The concept of over-fitting in MATLAB DANN toolbox is similar to the concept used in NN Toolbox in MATLAB® (Beale et al., 1992). The main idea is to terminate the RNN before the network gets over-fitted. For early stop, Mean Squared Error (MSE) for both training and validation is continuously monitored. While training a network, learning algorithm builds a certain hypothetical model for the network at each epoch.

The validation data are validated using the hypothetical model at that particular epoch. While learning, the value of MSE of training and validation data keep on reducing and the training of the network gets better with increasing epoch. However, the validation error can increase though the error for training decreases, which occurs in cases of over-fitting. The deterioration in the validation error can be attributed to the training process with random behavior (Beale et al., 1992). Therefore, the MATLAB DANN toolbox will count six consecutive increments in the validation error before it stops the learning algorithm.

When the model is over-fitted, the trained model seems to have good performance with training data, but it can have a large error while testing with the new data set (Beale et al., 1992). In other words, the trained model is not a generalized model when it is over-fitted. The implementation of validation check is presented in Case II in Section 3. In case, if the validation check is not selected, validation data will be part of the training data set.

2.1.4 Bias

It is an offset value added to the output of the neurons. It is often important to include bias in each neuron while constructing a neural network model. MATLAB DANN toolbox facilitates a choice to include or exclude bias terms in the network.

2.1.5 Learning algorithm

In this toolbox, there are three learning algorithms. The user can select any one of these algorithms based on the requirements regarding complexity, accuracy and application.

Back Propagation Through Time (BPTT) BPTT is an extension of gradient-based backpropagation algorithm that is used in feedforward ANN. The idea in BPTT is to unfold the RNN architecture into feedforward ANN architecture using an arbitrary number of folds. The BPTT architecture for the neural network with two neurons is shown in Figure 2. The network with BPTT algorithm is less complex compared to other learning algorithms. However, the complexity and the memory requirement increase when the number of folds increases. (Williams, 1992)

Real Time Recurrent Learning (RTRL) RTRL is one of most accepted real-time learning algorithms for RNN. In RTRL, the gradients at time 't' are computed using the propagation of gradients at previous time steps (Mak et al., 1999; Mandic and Chambers, 2000; Budik, 2006). The underlying RTRL architecture is shown in Figure 3,

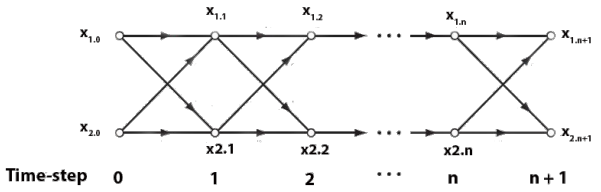

Figure 2. Architecture for Back Propagation Through Time (BPTT) learning algorithm with two neurons and n numbers of folding. (Haykin, 2009)

where **x**, **y**, N and d are inputs, outputs, number of neurons and unit time delay respectively. Based on the complexity, RTRL is the simplest online learning algorithm. However, the algorithm converges slowly and requires large memory for storage. (Williams, 1992)

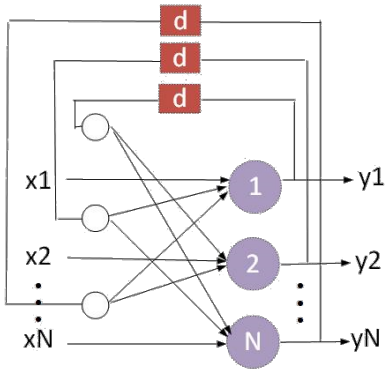

Figure 3. A general architecture for Real Time Recurrent Learning (RTRL) and Extended Kalman Filter (EKF) learning algorithms showing self-feedback and feedback loops within the neurons. Vectors **x** and **y** as input and output with d as the delay.

Extended Kalman Filter Learning (EKF) EKF can be used as a supervised on-line learning algorithm to tune the weights of RNN. In EKF, the state vector consists of weights and the locally induced outputs of each neuron in the network. Regarding convergence, EKF is the fastest algorithm among the algorithms presented in the MATLAB DANN toolbox. The order of computational complexity for EKF is the same as for RTRL, and the storage requirement is larger for EKF. The general architecture for EKF learning is shown in Figure 3. (Williams, 1992)

The main problem using gradient-based learning algorithms is vanishing gradient problem. As a solution to this problem, the German researcher Sepp Hochreiter and Juergen Schmidhuber introduced recurrent net with Long Short-Term Memory (LSTM) units (Haykin, 2009). In recent publications (Sak et al., 2014; Zen and Sak, 2015), LSTM RNN architectures are implemented because of their accuracy.

2.1.6 Learning parameters

The parameters of learning algorithms such as the number of neurons, learning rate, the maximum number of epochs and number of folds are discussed in this section.

Number of neurons The neurons and the connections between the neurons are essential features of a neural network. The number of neurons plays a vital role in the performance of the neural network. Too few neurons may not completely describe the dynamics of the system, and too many neurons can increase the complexity of the network (Haykin, 2009; Siddique and Adeli, 2013). Therefore, an optimal selection of a number of a neuron is one of the most important aspects of neural network modeling. In MATLAB DANN toolbox, each neuron is associated with the sigmoid function with the range [0, 1].

Learning rate The learning rate determines the rate of learning of gradient-based learning algorithms like BPTT and RTRL. The range of learning rate is [0, 1] and determines the converging efficiency while learning. The very small value of learning rate will slow down the learning algorithm and may require a large number of epochs to converge to a solution. Whereas the high value of learning rate can converge quickly, but it might have large variations and fluctuations in MSE of a training data. (Haykin, 2009; Siddique and Adeli, 2013)

Maximum number of epoch In MATLAB DANN toolbox, there are two stopping criterions. One of them is validation check, which is already discussed. Another way of stopping the training is the maximum number of epochs. A user can select a maximum number of epochs for the training using DANN Menu.

Number of folds The number of folds is a parameter for BPTT learning. The default selection is '3', which is the minimum possible value that can be selected for a given number of folds.

2.1.7 Past inputs and outputs

In applications involving prediction of time-series, the current output depends on the previous inputs and outputs. MATLAB DANN toolbox allows a user to select a number of previous inputs and previous outputs as additional inputs to find the output at the current time. By default, if the values are selected as '0' for both input and output, MATLAB DANN toolbox will use one previous output from each neuron.

2.1.8 Additional parameters

In EKF learning algorithm, a user must assign three more parameters for learning, which are Sigma_U, Sigma_W, and Sigma_O in MATLAB DANN toolbox. These parameters are tuning parameters for the output of each neuron as a state, weights as a state and output of the network respectively. These parameters are responsible for Kalman gain calculation for the states (i.e. output of neuron and weight) and determine the update of the output of each neuron and the weight connections between the neurons (Williams, 1992). As the simulation stops, the parameters, weights and other information regarding the simulation are saved in the workspace in MATLAB®.

2.2 Parameter Tuning

In any implementation of ANN, tuning of parameters is one of the biggest challenges. The optimal selection of network parameters can only lead to a good model. Contrary to existing MATLAB® Neural Network Toolbox, MATLAB DANN toolbox has a facility to tune the parameters optimally. In DANN Main, if you click on Tuning button, Parameter Tuning window will open as shown in Figure 4. The optimal tuning is based on the grid search method, and optimality is evaluated using Mean Absolute Percentage Error (MAPE). In the left panel of the window, a user can assign lower limit, higher limit and an increment to each parameter and start tuning. At the end of the tuning, optimal values of the parameters are displayed in the right panel of the window with minimum MAPE. Usually, parameter tuning takes a long time, so MATLAB DANN toolbox provides an option to get notification alarm. It is to be noted that a user must upload data, select learning algorithm, decide to or not to include bias and validation check before starting the tuning process. Thus, obtained optimal parameters can be used for training the model.

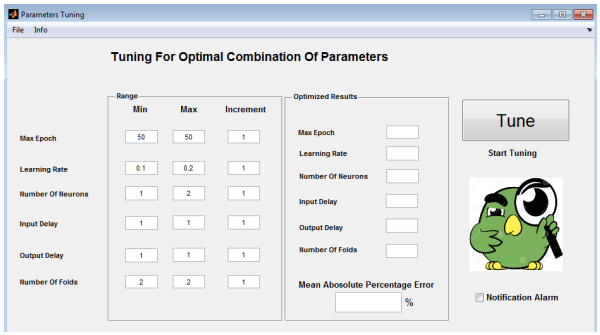

Figure 4. Parameter Tuning window of MATLAB DANN toolbox that allows a user to tune the optimal parameters based on grid search method.

2.3 Plot Menu

Plot Menu window pops-up when the simulation is completed as shown in Figure 5. It consists of five different types of plots, which are performance plot, regression plot, prediction plot, parameter plot, and error plot.

2.3.1 Performance plot

It shows the MSE for training data set and validation error for each epoch.

2.3.2 Regression plot

It compares the target output and model prediction in terms of squared correlation coefficient such that '0' meaning not related at all and '1' meaning highly correlated to each other.

2.3.3 Prediction plot

It shows the test data and model prediction with MAPE between them.

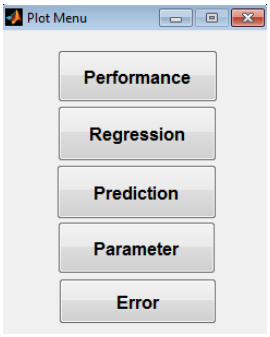

Figure 5. Plot Menu of MATLAB DANN toolbox with different plots for the analysis of the model.

2.3.4 Parameter plot

It shows the states of five different randomly chosen weights at different epochs. The analysis using parameter plot is very efficient if you are working with some system identification problems. In that case, one can visualize how the weights change with epochs. The steady state values of the weights after some epochs are the model parameters in typical system identification problems.

2.3.5 Error plot

It shows the error between the target value and the model prediction for each test samples.

2.4 Additional information

The MATLAB DANN toolbox has additional help options for the users. A user can get general information in Q&A section inside the Help Window. With a right-click in any parameter name, action buttons or selection options, a prompt help window related to that expression will pop up.

The MATLAB DANN toolbox in the current version has the following limitations:

(a) The toolbox is not yet ready for linking to Simulink.

(b) The toolbox handles Multiple Input Single Output (MISO) scenario and needs some modification to address Multiple Input Multiple Output (MIMO) scenarios.

(c) The execution time needed for the current version can be reduced.

2.5 Installing the MATLAB DANN toolbox

The first step in installing MATLAB DANN toolbox is to download installation file. Double-click in the downloaded installation file will direct to the installation process in the MATLAB®. It is recommended to have a MATLAB® version 2014 or later.

3 Case Studies

In this section, the usage of the MATLAB DANN toolbox in three different practical applications is discussed.

These three different cases use the data set from an experimental flow rig, and example data sets from MATLAB® Neural Network Toolbox. To give a better understanding in analyzing the simulation results, different sets of plots are investigated under these cases.

3.1 Case I: BPTT learning algorithm for flow measurement

In drilling operations, the flow rates of drilling mud at inflow and outflow positions can be used to detect kick and fluid loss. An open channel flow loop is available at University College of Southeast Norway (USN) for the study of outflow measurement. The data set with three level measurements as inputs and a flow measurement as the single output are taken from the flow loop for the analysis of BPTT learning algorithm in MATLAB DANN toolbox. Figure 6 and Figure 7 show the regression plot and prediction plot for flow estimation using BPTT learning algorithm in the toolbox. The simulation results show that the BPTT learning algorithm provided by MATLAB DANN toolbox is capable of mapping the inputs and outputs with high accuracy.

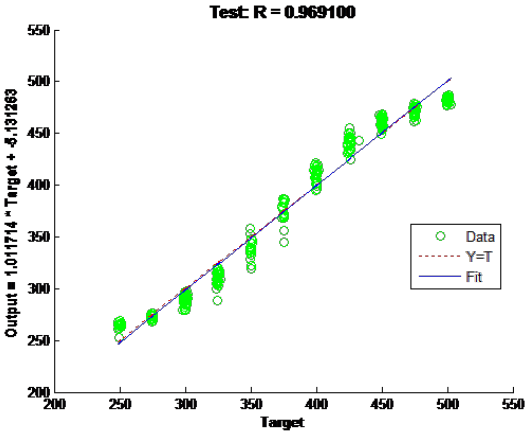

Figure 6. The regression plot for flow measurement using BPTT learning algorithm, with a correlation of 96% between the target values and the model prediction values. Data set from an experimental flow rig at USN.

3.2 Case II: EKF learning algorithm for temperature measurement

To analyze the performance of EKF learning algorithm, an example data set provided by MATLAB® Neural Network Toolbox is used. The data set of a liquid-saturated steam heat exchanger consists of time-series liquid flow rate and liquid outlet temperature, used as input and output to the ANN feedback network respectively. Figure 8 and Figure 9 show the performance plot and prediction plot for fully connected RNN with EKF learning algorithm. The learning algorithm has an early stop at 8 epochs due to the validation check with MSE of 0.015. The low value of MAPE in prediction plot shows that the EKF learning algorithm with validation check is able to generalize the model and avoid over-fitting.

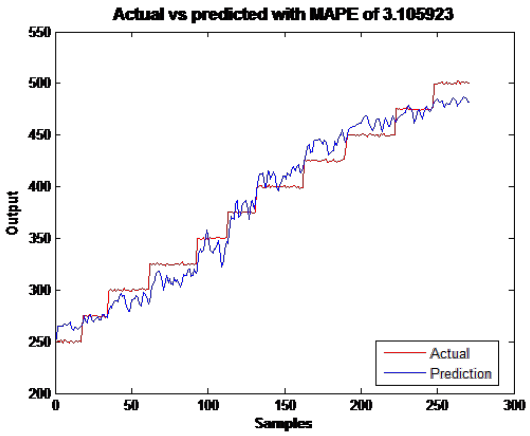

Figure 7. The prediction plot for flow measurement using BPTT learning algorithm with a MAPE of 3.1%. Data set from an experimental flow rig at USN.

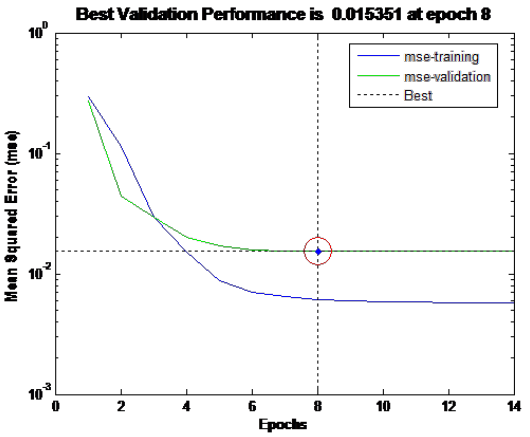

Figure 8. The performance plot for temperature measurement using EKF learning algorithm. The best validation performance is 0.015351 at epoch 8. Data set from MATLAB® Neural Network Toolbox.

3.3 Case III: RTRL learning algorithm for mortality prediction

Another example data set provided by MATLAB® Neural Network Toolbox is used to investigate the performance of RTRL learning algorithm. The data set is a Pollution mortality data set that consists of eight input variables (Temperature, Relative humidity, Carbon monoxide, Sulfur dioxide, Nitrogen dioxide, Hydrocarbons, Ozone, and Particulates) and total mortality as an output variable. Figure 10 to Figure 13 shows the simulation results for mortality prediction using RTRL learning algorithm using MATLAB DANN toolbox.

The parameter plot as shown in Figure 10 shows the states of randomly chosen weights of the network. As discussed in Section 2, it takes longer time for the weights to converge to a steady state when using RTRL.

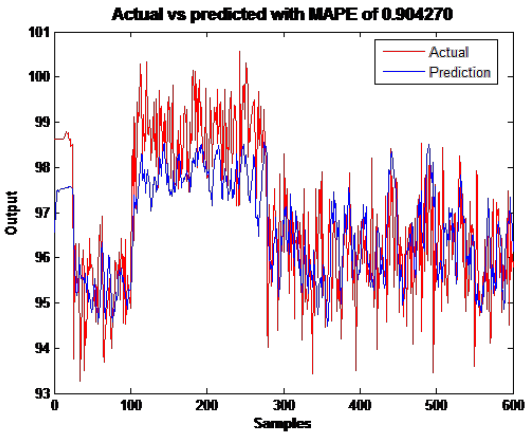

Figure 9. The prediction plot for temperature measurement using EKF learning algorithm with a MAPE of 0.9%. Data set from MATLAB® Neural Network Toolbox.

The regression plot as in Figure 11 illustrates that the predictions using RTRL are highly correlated with the target values with a correlation of 92%. The MAPE between the predicted values and target values are 2.93% as shown in Figure 12. The error in each sample is shown in the error plot in Figure 13.

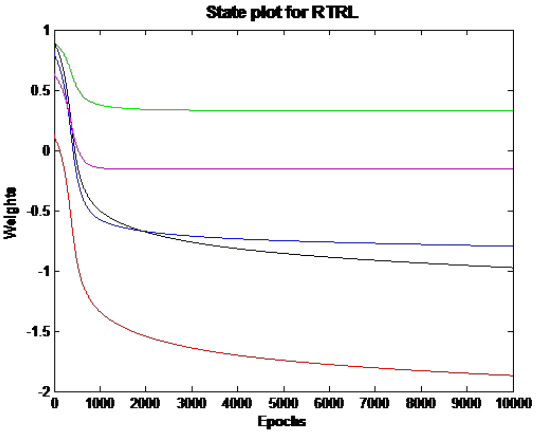

Figure 10. The state plot for mortality time-series prediction using RTRL learning algorithm. Data set from MATLAB® Neural Network Toolbox.

4 Conclusions

The existing MATLAB® Neural Network Toolbox has a possibility to use both static and dynamic neural networks. However, it is not possible directly use the toolbox to fully connected recurrent neural networks. For this reason, this study presents the Dynamic Artificial Neural Network MATLAB toolbox that gives an opportunity to use the fully connected neural network for time-series predictions. The toolbox consists of three different learning algorithms, where Back Propagation Through Time (BPTT) is an offline learning algorithm, Real Time Recurrent Learning (RTRL) and Extended Kalman Filter (EKF)

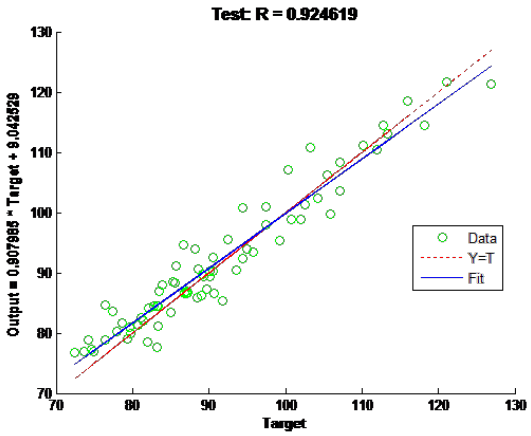

Figure 11. The regression plot for mortality time-series prediction using RTRL learning algorithm with 92% correlation between target values and model predictions. Data set from MATLAB® Neural Network Toolbox.

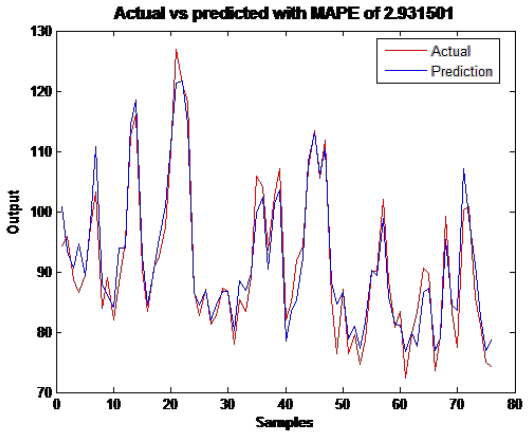

Figure 12. The prediction plot for mortality time-series prediction using RTRL learning algorithm with MAPE of 2.93%. Data set from MATLAB® Neural Network Toolbox.

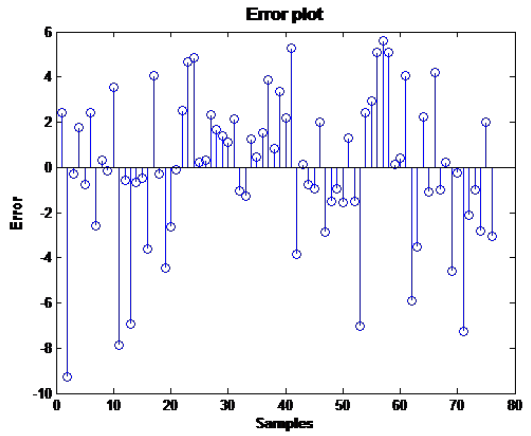

Figure 13. The error plot for mortality time-series prediction using RTRL learning algorithm with 9 units as the highest error in the test samples. Data set from MATLAB® Neural Network Toolbox.

learning algorithm are online learning algorithms. Main details and guides for installing and using the developed toolbox are presented in this paper.

To demonstrate the features of the MATLAB DANN toolbox, three different practical problems are considered using three different learning algorithms. The simulation studies presented in this paper show that the developed toolbox can be used in applications involving time-series predictions. In addition, the developed toolbox has a dedicated option for the optimal tuning of parameters.

This Toolbox can be used in financial market trending studies with some modification similar to (Pelusi et al., 2014).

This work is meant for academic use with particular focus on the students using the existing MATLAB® Neural Network Toolbox. In future, other different learning algorithms can be included in the developed toolbox with some programming efforts.

Acknowledgement

The Ministry of Education and Research of the Norwegian Government is funding Khim Chhantyal's PhD studies at University College of Southeast Norway (USN). Minh Hoang was partly involved in the development of this paper in conjunction with his master thesis at USN in 2016, (Hoang, 2016). The authors at USN appreciate the collaboration with and support from STATOIL for the rig used in the current studies for generation of time series of flow rates. We appreciate the expert advice on drilling operations by Dr. Geir Elseth of STATOIL. In addition, we acknowledge the practical work done by various groups of bachelor and master students of USN in conjunction with this project. Part of the work done is associated with the project SEMI-KIDD funded by the Research Council of Norway.

References

Daniel Borisovich Budik. *A Resource Efficient Localized Recurrent Neural Network Architecture and Learning Algorithm*. Master Thesis, University of Tennessee, 2006.

Danilo P Mandic and Jonathon A Chambers. A normalised real time recurrent learning algorithm. *Signal processing*, 80(9):1909–1916, 2000. doi:10.1016/S0165-1684(00)00101-8.

Danilo Pelusi, Massimo Tivegna, and Pierluigi Ippoliti. Intelligent algorithms for trading the euro-dollar in the foreign exchange market. In *Mathematical and Statistical Methods for Actuarial Sciences and Finance*, pages 243–252. Springer, 2014. doi:10.1007/978-3-319-02499-822.

Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Heiga Zen and Hasim Sak. Undirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on, pages 4470-4474. IEEE, 2015. doi: 10.1109/ICASSP.2015.7178816.

r LPJ Veelenturf and Ir SH Gerez. Analysis of recurrent neural networks with application to speaker independent phoneme recognition. 1999.

Man-Wai Mak, Kim-Wing Ku, and Yee-Ling Lu. On the improvement of the real time recurrent learning algorithm for recurrent neural networks. *Neurocomputing*, 24(1):13–36, 1999. doi: 10.1016/S0925-2312(98)00089-7.

Mark Hudson Beale, Martin T Hagan, and Howard B Demuth. *Neural Network Toolbox^{TM} User's Guide*. 1992.

Minh Hoang. *Viscosity measurement of non-Newtonian fluids*. Master Thesis, University of South East Norway, Norway, 2016.

Nazmul Siddique and Hojjat Adeli. *Computational intelligence: synergies of fuzzy logic, neural networks and evolutionary computing*. John Wiley & Sons, 2013.

Ronald J Williams. *Some observations on the use of the extended Kalman filter as a recurrent network learning algorithm*. College of Computer Science, Northeastern University, 1992.

Simon Haykin. *Neural networks and learning machines*, volume 3. Pearson Upper Saddle River, NJ, USA, 2009.