Solution to ARGESIM Benchmark C18 'Identification of Nonlinear Dynamical Relations' using Excel with a Neural Networks Add-In

Sašo Blažič, University of Ljubljana, Slovenia; *saso.blazic@fe.uni-lj.si*

Simulator: It is known for quite some time that artificial neural networks (ANNs) are capable of modelling very complex nonlinear systems. Hence, they have become very popular and widely used in a variety of applications. A reasonable conclusion was that ANNs should be somehow accessible to a wider public. Many companies started developing Add-Ins for Microsoft Excel that has become the industry-standard data analysis and modelling tool. These Add-Ins are usually very easy to use and a user without much experience with ANNs is capable of training a network and using it for prediction or classification purposes. The drawback is that the user is very limited in customising the ANN.

Two tasks (identification with parallel structure of a linear model and an ANN, and identification with a dynamical ANN) were solved in the paper with the following software configuration: Microsoft Office Excel 2003, and NeuralTools, Neural Net Add-In for Microsoft Excel, Version 1.1.0 – Professional Edition, Palisade Corporation.

Model: When using NeuralTools, neural networks are developed and used in four steps:

- *Data preparation*. The data used in NeuralTools are defined in data sets. A Data Set Manager is used to set up data sets so they can be used over and over again with neural networks.
- *Training*. With training, a neural network is generated from a data set comprised of cases (input vectors) with known output values. This data often consists of historical cases for which the values of output/dependent variable are known.
- *Testing*. With testing, a trained neural network is tested to see how well it does at predicting known output values. The data used for testing is usually a subset of historical data. This subset was not used in training the network. After testing, the performance of the network is measured by statistics such as the percentage of the known answers it correctly predicted.
- *Prediction*. A trained neural network is used to predict unknown output values. Once trained and tested, the network can be used as needed to predict outputs for new case data.

Each of the steps is done by simply clicking the appropriate icon and setting a few parameters.

NeuralTools supports different neural network configurations to give the best possible predictions. For classification/category prediction (where the dependent variable is a category type), two types of networks are available: Probabilistic Neural Networks (PNN) and Multi-Layer Feedforward Networks (MLFN). Numeric prediction can be performed using MLFN networks, as well as Generalized Regression Neural Networks (GRNN), which are closely related to PNN networks.

NeuralTools makes selecting a network configuration easy by offering a Best Net search. When selected, NeuralTools will train and test a variety of neural network configurations to generate the one that gives the best predictions for the data. The best configuration is determined based on testing data.

A The linear model used is the same as the one proposed in the solution by Aleš Belič, identified in MATLAB, and not separately investigeated in EXCEL:

$$G(z) = \frac{y_{lin}}{u} = \frac{0.5289z^2 - 0.5206z}{z^2 - 1.586z^1 + 0.5985} \qquad (1)$$

There y_{lin} is the output of the linear part, with input variable u (thickening) and output variable y (force).

B-Task: In this task the parallel connection of an LTI system and a static ANN was used. The linear model is given in (1). The output of the model is $y_{lin} + y_{ANN}$, and the ANN is therefore trained with u at its output and the residual error $y - y_{lin}$ at its output.

Two columns are prepared in Excel. In the first column the input samples (from the identification signal) are prepared and in the second column comprises of the samples of $y - y_{lin}$. These two signals are previously prepared in MATLAB and imported into Excel. Then (identification) data set is defined with the Data Set Manager. The training is started with the Generalized Regression Neural Networks that does not require any designer parameters (whereas Multi-Layer Feedforward Networks require the number of nodes to be defined). (Actually, this is a very simple map-

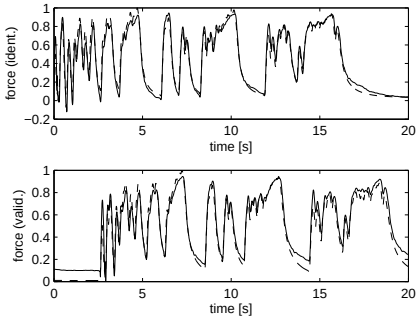

Figure 1. Simulation of the parallel structure of linear dynamical model and the ANN (solid line) in comparison with the measured force (dashed line). Training data set (top), validation data set (bottom).

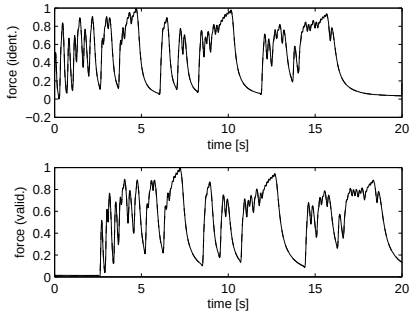

Figure 2. One-step-ahead prediction of the neural network model (solid line) in comparison with the measured force (broken line). Training data set – above, validation data set – below.

ping with one input and one output, and therefore any setting results in a similar network response.)

The ANN and the linear dynamical model are then simulated in parallel and the result of the hybrid system is shown in Figure 1. Calculated correlation coefficients for this case are 0.9805 and 0.9814 for the training and the validation data set, respectively.

C-Task: In this task the dynamical ANN is used. There are four inputs to the system: $y(k\text{-}1)$, $y(k\text{-}2)$, $u(k\text{-}1)$, and $u(k\text{-}2)$, and the output is $y(k)$. The ANN is trained on the identification data. The network response is simulated in the validation phase (with the new data).

NeuralTools does not enable training of dynamical neural networks directly. In the training phase only a static data set is permitted. In the validation phase we want to simulate the response of the ANN to a particular input signal which is again not possible directly with the NeuralTools. The software only offers the possibility of predicting the output to a static data set. In order to simulate a dynamical network one needs to feed back the delayed network outputs. But NeuralTools has a special feature that can be made good use of in order to simulate the dynamical network. It is possible to start the prediction in the "Live prediction" mode. This means that the output of the network is recalculated as soon as the input data change (in Excel worksheet). A simple Excel macro was written that iteratively copies network predictions and pastes them as inputs for the future outputs. Thus, outputs are effectively fed back and the network becomes dynamical.

NeuralTools with the addition of a simple macro are therefore capable of simulating the dynamical ANN. The problem is that in the training phase it is virtually impossible to achieve a stable dynamical ANN for this case. A vast amount of tests were made and the results were always the same: very good prediction ability (for identification and validation data) and locally unstable models when using feedback. It is well-known that the dynamical neural networks are much harder to train than the static ones. Although they can be trained using the same gradient-based algorithms that are used for static networks, the performance of the algorithms on dynamic networks can be much worse. If the simulation of a dynamical ANN is possible with some extensions, the training procedures turned out not to be suitable for training of the dynamical ANN for this case.

Since unusable simulation data were obtained, one-step-ahead predictions are shown in Figure 2.

Resumé: NeuralTools turned out to be quite effective in training static networks. The software is intuitive, very simple to use and suitable also for users without much experience and knowledge about artificial neural networks. This simplicity is also a certain drawback since an experienced user is very limited in customising the network. It turned out that the training algorithm of the software is not very suitable for dynamical networks and similar problems as the ones faced in our experiments can be encountered. It is true that the data used in this case are difficult for the identification since a slight change of the dynamics with time can be observed and also the input signal excitation is not very good at high frequencies.

Corresponding author: Sašo Blažič,
Univ. of Ljubljana, Faculty of Electrical Engineering,
Tržaška 25, 1000 Ljubljana, Slovenia
saso.blazic@fe.uni-lj.si

Received: September 14, 2007
Revised October 10, 2007
Accepted October 15, 2007