Using the neural net in time-series forecasting

Jens Mende and Steve Banhegyi

ABSTRACT

Data Engine's multi-layer perceptron was incorporated into a forecasting system together with the classical methods of moving averages, exponential smoothing, and linear regression. When the system was tested with sine waves, the perceptron proved to be far superior to the classical methods, particularly in forecasting several days into the future. Subsequently the perceptron predicted price indices more accurately than the classical methods. When leading indicators were included in its training examples, the perceptron even predicted the prices of individual shares more accurately than the classical methods

Jens Mende,Department of Information Systems, University of the Witwatersrand, Private Bag 3, Wits 2050, South Africa. jensm@isys.wits.ac.za

Steve Banhegyi,Ex-Teq Asset Management, PO Box 413176, Craighall, 2024, South Africa. steve@ex-teq.co.za

INTRODUCTION

Planners in many walks of life are continually faced with the need to forecast the future value of a time series. For example, a marketing manager needs to forecast next year's sales from a firm's sales during the past five years. Similarly a town planner needs to forecast next month's demand for municipal services from the demand during the previous sixty months. And an investment analyst needs to forecast the future value of a share price from the prices during the previous thousand days.

For these purposes, planners have long been using classical forecasting methods such as *moving averages*, *exponential smoothing* and *linear regression*. More recently, a new forecasting tool has been made available by firms such as MIT, namely the *neural net*.

The authors have developed a forecasting system which inputs a time series, applies the classical methods as well as a neural net, and outputs comparative forecasts. This paper outlines the system's processing functions and compares the relative predictive success of the various methods.

THE NEURAL NET

The core of the system is a neural net, namely the DataEngine multi-layer perceptron (MLP). The MLP is a general-purpose forecasting tool that can be trained to predict any system's output variables from its input variables, i.e.

$$[x_1 \;\; x_2 \ldots x_n] \;\rightarrow\; [y_1 \;\; y_2 \ldots y_m].$$

For example, in the case of a motor-car carburettor, the input variables x_1 and x_2 might be engine speed and temperature, and the output variables y_1 and y_2 might be air flow and fuel flow. In training mode, one would give the MLP a set of training examples that include actual measurements of the input variables and the corresponding output variables, for example:

$$[10, 10] \;\rightarrow\; [5, 5]$$

$$[10, 12] \;\rightarrow\; [5, 6]$$

$$[12, 10] \;\rightarrow\; [6, 5]$$

$$[12, 12] \;\rightarrow\; [6, 6].$$

The MLP will then recognise the pattern in the input / output relationship. Subsequently, in prediction mode, one could then supply measurements of the input variables, and it will use that pattern to predict the corresponding output values. For example, from input values such as [11, 11], it might predict output values such as [5.5, 5.5].

In the case of a time series, the MLP can be used in much the same way. Now the input variables are past values of the time series, and the output variables are future values, for example:

$$[1, 2, 3, 4, 5, 6, 7] \;\rightarrow\; [8, 9].$$

In training mode, the MLP again requires a set of training examples. However the situation is more complicated here. One cannot include all the past and future values in every training example, because then there would only be one training example, and the MLP would be unable to recognise the pattern. In order to generate many training examples, time is therefore divided into slices, and in each time-slice the input variables are associated with the past, while the output variables are associated with the future. So a given time series such as 1, 2, 3, 4, 5, 6, 7 might be divided into three time-slices: 3, 4, 5, 6, 7; 2, 3, 4, 5, 6; 1, 2, 3, 4, 5. These time-slices would yield three training examples:

$$[3, 4, 5] \rightarrow [6, 7]$$

$$[2, 3, 4] \rightarrow [5, 6]$$

$$[1, 2, 3] \rightarrow [4, 5].$$

In prediction mode, one can then feed the past values of the most recent time-slice into the MLP, and it will predict the future portion of that time-slice … for example:

$$[5, 6, 7] \rightarrow [8, 9].$$

TIME-SLICING

Slicing a time series is a laborious task, particularly if the time series is long and the slices are short. So this job needs to be automated. However, DataEngine does not provide a time-slicing function. Therefore the authors have devised an Excel pre-processor that divides a given time series into training examples.

The pre-processor inputs spreadsheet A which contains a time series, and outputs spreadsheet B which contains training examples. It begins by finding a user-supplied time series in the first column of spreadsheet A, and counts the number of rows in that column (Trows). Next, it reads the time series into a memory array Tval(Trows). Then it generates training examples in the rows of spreadsheet B.

The generator procedure (in VBA) is remarkably simple:

```
Npast = 15                          'Each training example should include 15 past values
Npred = 5                           'Each training example should include 5 future values
Nslice = Npast + Npred              'Number of values in each time-slice
Negs = Trows - Nslice +1            'Number of training examples to be generated
For Neg = 1 To Negs                 'Repeat for each training example
    Wrow = 2 + Neg                  'Store examples from worksheet row  3 onwards
    Wcol = 0                        'Start at worksheet column 0

    Trow9 = Trows - (Neg - 1)       'Last time-series row to be included in the example
    Trow1 = Trow9 - Nslice + 1      'First time-series row to be included in the example
    For Trow = Trow1 To Trow9       'Select each row to be drawn from the array
```

```
        Wcol = Wcol +                    'Select the corresponding column in the worksheet
1

        WS.Cells(Wrow, Wcol) = Tval(Trow)    'Store the series value in the worksheet

                                         'Repeat for the next row of the time series array
Next

Next                                     'Repeat for the next training example
```

A similar procedure also derives the past values of the most recent time-slice from the array, and stores them in spreadsheet C. A DataEngine card then converts spreadsheets B and C into DataEngine format so that the user can run the MLP in training (or retraining) mode. Another DataEngine card then runs the MLP in recall mode, and stores the predicted future values of the time series in spreadsheet D. Finally an Excel post-processor plots the future values on a graph together with the most recent past values.

AVERAGING

The pre-processor contains further procedures for deriving classical predictions from the time series array. The first of these procedures draws a regression line through the most recent values of the time series, and then predicts future values of the time series by extrapolating the line. The second procedure calculates four moving averages whose periods are specified by the user, for example 5-day, 10-day, 15-day and 40-day moving averages. The third procedure calculates two exponentially-smoothed averages whose weighting factors are also specified by the user, for example a smoothed average where the most recent value is weighted by 80%, and another smoothed average where the most recent value is weighted by 20%. The first procedure then derives additional time series predictions from these averages by drawing regression lines through them.

If the user requires the pre-processor to execute these procedures, then the post-processor plots the regression predictions together with the neural predictions on the same chart, so that the user can compare them.

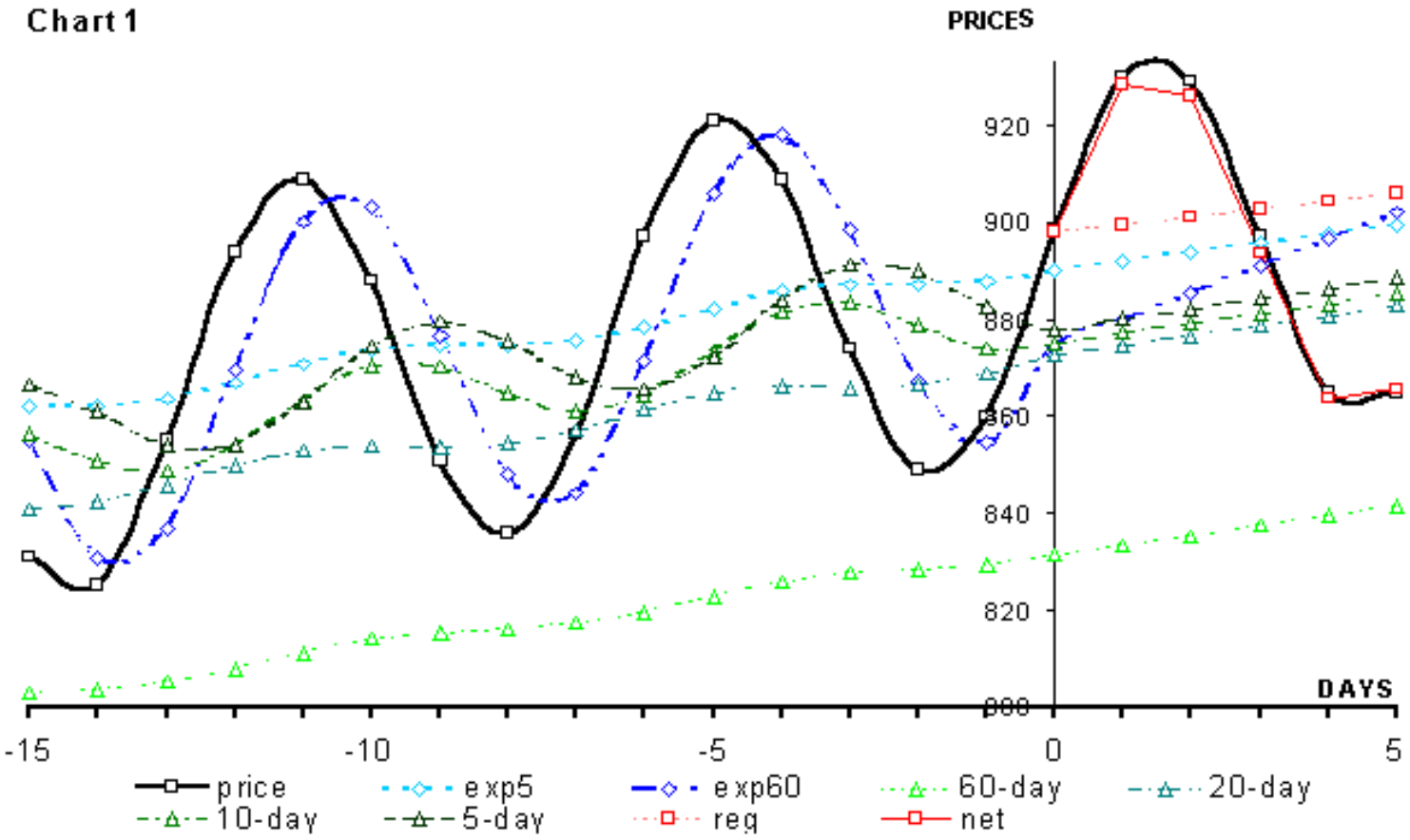
The pre-processor also contains a fourth procedure that calculates deviations of the time series from a specified series of averages. If this procedure is execute, it lets the generator procedure include deviations in each training example instead of time series values. Consequently the neural net will predict future deviations instead of future time series values. Accordingly the post-processor will add the neural-predicted deviations to the regression-predicted averages before and plotting the chart.

In summary, then, the MLP and its pre- and post-processors produce a chart which shows:

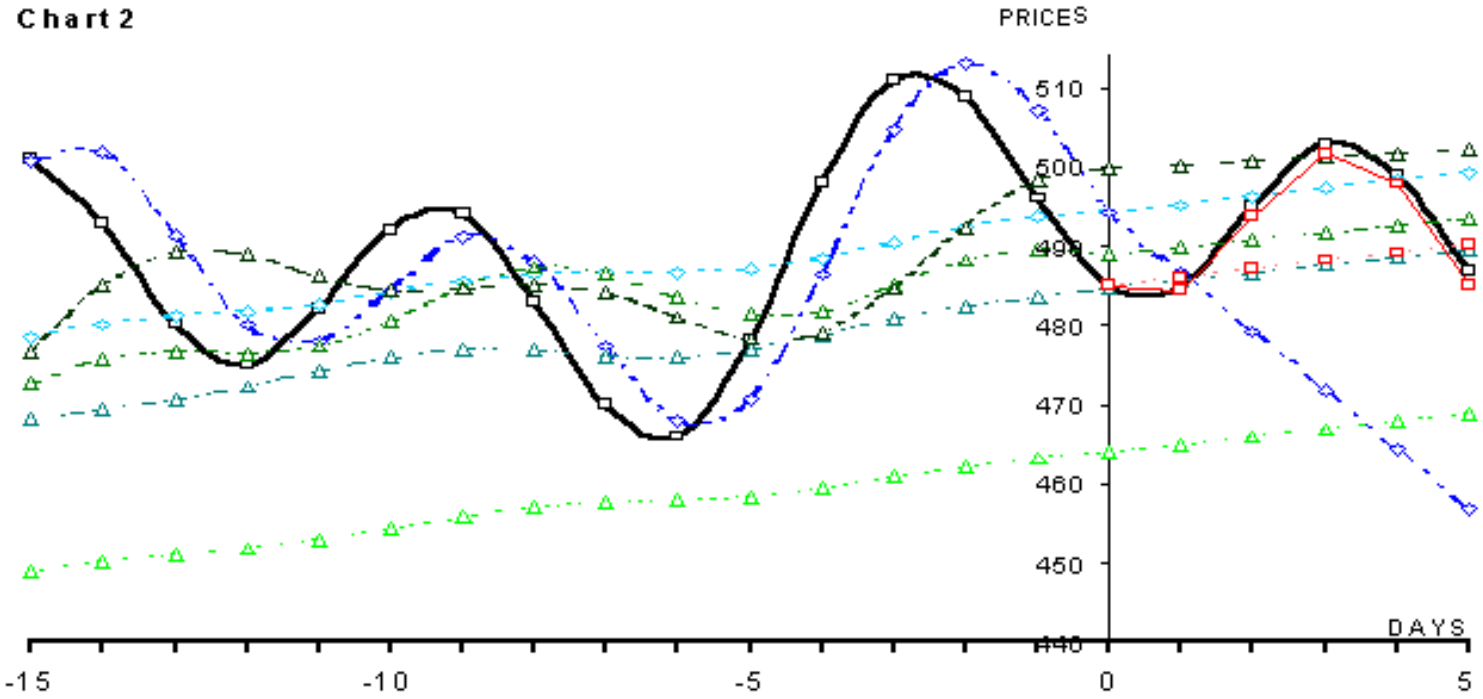
- the most recent past values of the time series, and regression predictions of those values
- moving and smoothed averages of the time series, and regression predictions of those averages
- neural predictions of the time series, either directly from series values or indirectly via deviations.

COMPARATIVE TESTS

The authors have tested this system with several artificial time series. One of these was a sine wave that oscillated about an upward-sloping straight line. After training with deviations from 99-day moving averages, the neural net predicted the next five future values. Chart 1 below shows that these predictions were very much more accurate than the corresponding predictions produced with the classical methods.



Another test case involved a sum of three sine waves that oscillated at different frequencies. Chart 2 below shows that the neural net again predicted this time series very much more accurately than the classical methods. In the light of Fourier's theorem - which proves that many functions can be decomposed into a series of sine waves - this result suggests that a neural net could potentially be used to forecast many functions.



COMPARATIVE FORECASTS

After testing, the authors used this system to forecast several real-life time series. One of these was the daily closing value of the all-shares index of the Johannesburg Stock Exchange (ALSI). Charts 3 and 4 show that the neural net sometimes predicted this price index more accurately than the classical methods, and sometimes less accurately.

LEADING INDICATORS

Approximately half of the neural net forecasts were as inaccurate as chart 4. This indicated that the future values of a time series depends not only on the pattern of its past values, but on other factors too. Accordingly, spreadsheet A was extended to include several secondary time series that might serve as leading indicators of the primary series, and correspondingly the pre-processor was extended so that it would include in each training example not only a sample of values drawn from the primary series but also a sample from each secondary series. The extended system was used to re-forecast. Charts 5 and 6 show the new predictions that replaced Charts 3 and 4. Evidently the inclusion of leading indicators improved the forecasts significantly.

CONCLUSION

What has been accomplished in this paper?

First, if a general-purpose neural net is to be used for time series forecasting, then it can and should be

supplemented by a pre-processor that automates the job of time slicing.

Second, a neural net is much better at predicting a regular time series than the classical methods of moving averages, exponential smoothing and linear regression.

Third, a neural net can predict a time series more accurately if leading indicators are included in its training examples.

Overall, a neural net is a valuable tool for time series forecasting, and planners should consider using it.