Nonstationarity and Data Preprocessing for Neural Network Predictions of an Economic Time Series

Francesco Virili[1] and Bernd Freisleben[2]

[1]Department of Business Information Systems, University of Siegen, Hölderlinstr. 3, D–57068 Siegen, Germany, E-Mail: francesco.virili@acm.org

[2]Department of Electrical Engineering & Computer Science, University of Siegen, Hölderlinstr. 3, D–57068 Siegen, Germany, E-Mail: freisleb@informatik.uni-siegen.de

Abstract

The presence of stochastic or deterministic trends in economic time series can be a major obstacle for producing satisfactory predictions with neural networks. In this paper, we demonstrate the effects of nonstationarity on neural network predictions using the time series of the mortgage loans purchased in the Netherlands. We present different preprocessing techniques for removing nonstationarity, and evaluate their properties by producing multi-step predictions using a linear stochastic forecasting model and a neural network. The results indicate that detecting nonstationarity and selecting an appropriate preprocessing technique is highly beneficial for improving the prediction quality.

1 Introduction

Theoretically, universal nonlinear approximators such as feed-forward neural networks should be able to model nonstationary processes without requiring any preliminary transformation. In many practical time series forecasting applications of neural networks, however, nonstationarity, i.e. the presence of stochastic or deterministic trends in the time series, is a major reason for severely degrading the prediction performance. Thus, quite often it is highly advisable to remove nonstationarity from the time series by appropriate preprocessing techniques before a neural network is applied. In this paper, we discuss possible effects of nonstationarity on time series predictions and present several preprocessing techniques to deal with nonstationarity. Using the time series of the mortgage loans purchased in the Netherlands from January 1985 until December 1997, we demonstrate that both a linear and a neural network network predictor exhibit improved performances when nonstationarity is removed before model application. The paper is organized as follows. Section 2 discusses the basic issues of stationarity in time series. Section 3 presents possible effects of nonstationarity on neural network predictions. In section 4, experimental results for the mortgage loan time series are presented. Section 5 concludes the paper and outlines areas of future research.

2 Stationarity and Trends

Roughly speaking, a times series is stationary when its generating process is time invariant. In a formal definition of stationarity, we distinguish between complete stationarity and weak (or second-order) stationarity (see e.g. [12], page 9): if x_t $(t = 1, ..., N)$ is a time series, $x_{t,m}$ is a section x_{t-j} $(j = 0, ..., m-1)$ containing m consecutive terms of x_t and $p_{t,m}(x)$ is the distribution function of $x_{t,m}$, then x_t is said to be completely stationary if $p_{t,m}(x)$ is not a function of t for every finite m; x_t shows a weak stationarity when its mean $E(x_t)$ and variance $Var(x_t)$ are constant and its autocovariances $cov(x_{t+k}, x_t) = E[(x_t - E(x_t))(x_{t+k} - E(x_{t+k}))]$ are only depending on k and not on time t. Some kinds of nonstationarity can be detected by visual observation: for example, time-dependent changes in mean are often evidenced by a trend in the series, whereas a trend in variance is usually associated with an increasing/decreasing spread of the time series. Knowledge

0-7695-0619-4/00 $10.00

about the presence and characteristics of an eventual underlying trend is important to convert a nonstationary time series into a stationary one. If a nonstationary time series x_t can actually be transformed into a stationary series $\Delta^d x_t$ by differencing it d times, it is said to be integrated of order d, conventionally denoted as $x_t \sim I(d)$. As explained in [8], chapter 4, an integrated time series shows a special kind of trend which is called 'stochastic trend' (ST, equation (1)), as opposed to the so called 'deterministic trend' (DT, equation (2)). DT is just a linear function of time t plus a random disturbance: after n time steps, y_n is the summation of n times the intercept μ, of n times δt and of the last disturbance ϵ_n. The stochastic trend, instead, presents an accumulation of the past shocks: y_n is the summation of n times the drift δ and of the past shocks $\sum_n \epsilon$.

$$ST \quad y_t = \delta + y_{t-1} + \epsilon_t \tag{1}$$

$$DT \quad y_t = \mu + \delta t + \epsilon_t \tag{2}$$

To achieve stationarity in the presence of a deterministic trend, the appropriate preprocessing technique is the subtraction of the linear time trend, which is obtained by a regression on time. In the presence of a stochastic trend, the appropriate preprocessing technique is to calculate the first differences. A well known test for the detection of the appropriate trend is the Said-Dickey version of the Augmented Dickey-Fuller (ADF) test, which is explained in detail in [2], chapter 4 and in [8], chapter 4. The seasonal version of the ADF test is due to Dickey, Hasza and Fuller [6] and is usually denoted as the DHF test.

3 Nonstationarity and Neural Network Models

To the best of our knowledge, there are no theoretical reasons for invalidating neural network models built with nonstationary time series, and in fact many successful neural network approaches for financial forecasting applications did not explicitly address it (see e.g. [23] [16]); other authors did it in an indirect way, i.e. separating the trend and volatility components [22] or making a direct comparison with ARIMA models, with a log differences preprocessing [9]. In most cases, it is not clear to which extent and why a nonstationary time series can be somewhat troublesome for a neural network. In the following, there are some of the factors to take into account.

3.1 Data Set Normalization

Modelling, for example, the mortgage loan times series shown in Figure 1 with a neural predictor, the presence of a trend may have undesired effects on the data set normalization. Suppose that in 1993 we are using the available data set until December 1993 to obtain predictions for 1994, the predicted value for December 1994 should be around 7000, but a neural network will not be able to predict it because it is out of the range used for the normalization. Similar problems are encountered in multiple inputs models: if the input time series are not stationary in mean and the prediction input set is not available at training time, there is a good chance that the prediction inputs will be out of the normalization range.

3.2 Noise/Nonstationarity Trade-Off

Hatanaka [11] pointed out that the outcome of the stationarity tests on the US economic (Nelson-Plosser) time series is different for different time windows: the shortest post-war series are much easier to analyze and interpret than the full historical record, which includes the first part of the century. The so called structural change (or break) in a deterministic trend was introduced by [15] in the form of a change in the trend line intercept and slope, justified by the effect of the oil price shock in the seventies; more recently Bierens [3] proposed a unit root test based on a nonlinear deterministic trend, approximated by time polynomials.

The presence of structural changes can be regarded as a kind of nonstationarity, with a more complex time dependency than a simple linear trend, and together with the presence of noise it can be also problematic for neural network univariate time series models: the training on older data sets (long training window) can induce biases in the predictions because of nonstationarity, where using a shorter training window can produce estimation error (too much model variance) because of the noise in the limited data set. Moody called this phenomenon the noise/nonstationarity trade-off [13] and suggested to use the test error against

the training window length for the choice of the optimal training window. In the presence of borderline unit root time series and/or of nonlinear deterministic trends, the choice of the preprocessing is not always straightforward, and it is advisable to use all the available knowledge about the target time series and to compare the outcomes of different preprocessing methods.

3.3 Extrapolation and the Bias-Variance Dilemma

As stated by Geman, Bienenstock and Doursat [10], the expected error of the estimator can be decomposed into a 'variance' component which grows with the model complexity, and a 'bias' component which decreases with the model complexity. The right balance between bias and variance minimizes the total expected error, and it may be obtained using a number of free parameters (complexity) that is appropriate for the (unknown) complexity of the Data Generating Process (DGP). Even if we were able to choose the optimal dimensionality of the nonlinear estimator without knowing the DGP, we have to notice with the authors (page 44) that "... if a difficult classification task is indeed to be learned from examples by a general-purpose machine that takes as inputs raw unprocessed data, then this machine will have to "extrapolate", that is, *generalize in a very nontrivial sense*, since the training data will never "cover" the space of all possible input". The suggestion of the authors is then based on explicitly introducing some structure (based on knowledge of the DGP) in the model (page 45): "It would appear, then, that the only way to avoid having densely cover the input space with training examples - which is unfeasible in practice - is to *prewire* the important generalizations". Usually a powerful way to do this is data representation: for example, they report a successful application in a handwritten numerals recognition problem: after changing the data representation from the 'raw' pixel-array to the Hamming distance, they report: (pages 50-51)"...we have introduced a very significant bias, thereby achieving a better control of variance. We believe, more generally, that adopting an appropriate data representation is an efficient means for designing the bias required for solving many hard problems in machine perception". Similarly, in univariate time series prediction, the detection and the discovery of the characteristics of nonstationarity, with the consequent choice of the preprocessing, could be a way to eliminate the need to extrapolate and to introduce some knowledge of the DGP structure, presenting the estimator a more meaningful representation of the data.

4 Experimental Results

Many banks are interested in modelling the mortgage loans demand in order to plan the resources to be allocated to fulfill this demand and to reduce the related risk and costs. The monthly mortgage loans production time series for the Netherlands is depicted in Figure 1, for the period January 1985 until December 1997 (source: ABN-Amro Bank). The curve tracks down the total amount of new mortgage loans issued every month in the Netherlands to families, at the national level; the typical loan destination is housing. In our experiment, we are focusing on univariate models to obtain multi-step predictions up to 30 months ahead. We suppose to be in December 1994, just before the beginning of a new upward trend, and we try to extrapolate up to June 1997, using only the portion of the data set available until December 1994. We use two classes of univariate predictors: linear dynamic models (seasonal ARIMA (SARIMA): [4], chapter 9) and feed-forward backpropagation neural networks ([5], chapter 4).

4.1 Preprocessing Methods and Evaluation

The analysis of the distribution over time and spectral analysis (see [20] for details) suggested to take into account the logarithmic transformation, which is useful to correct the skewed distribution, and pointed out the presence of a probably stochastic seasonal pattern. The outcome of the ADF and DHF tests, shown in Figure 2, suggested the presence of a seasonal unit root, maybe together with a nonseasonal unit root. The tests were performed without including the hold-out sample (see below), using the SAS/ETS software package [18]. Accordingly, we decided to evaluate the following preprocessing methods: raw values; logs; logs + first differences (unit root) ; logs + seasonal differences (seasonal unit root); logs + first differences + seasonal differences (seasonal + nonseasonal unit root). To evaluate the out of sample performances of the multi-step predictions we used the normalised RMS error (NRMS), i.e. the Root Mean Squared Error divided by the standard deviation, of the target data set:

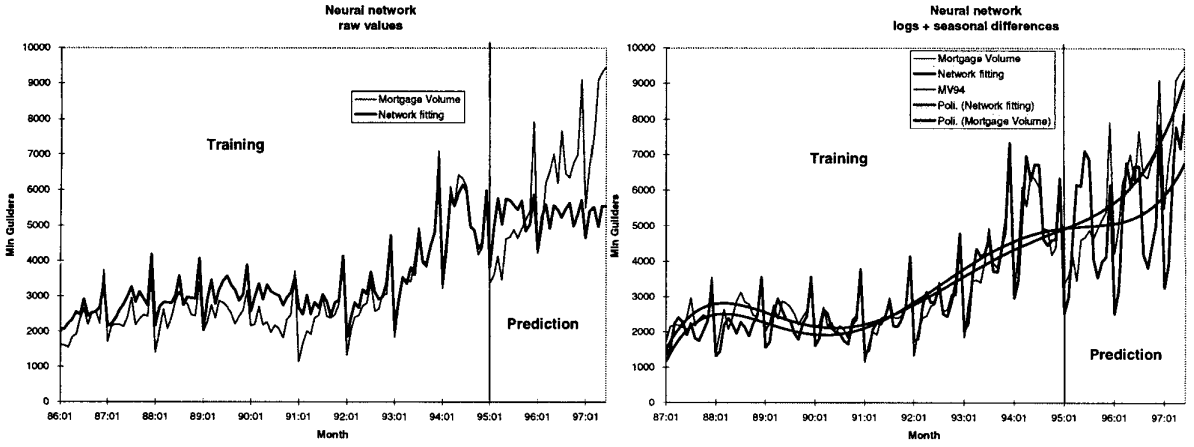

Figure 1: Neural network out of sample 30-step predictions, using raw values (left) or after preprocessing with log seasonal differences (right).

$$NRMS = \sqrt{\frac{\frac{1}{m}\sum_{i=1}^{m}(A(i) - P(i))^2}{(\frac{1}{n}\sum_{i=1}^{n}(A(i) - \bar{A}))^2}} \tag{3}$$

where A stands for actual values, P for Predicted values, m is the number of predicted patterns and n is the total number of patterns available at prediction time (until 12/94). An index below 1 is an indication of an improvement over the naive prediction obtained using the average. The perfect prediction would give an NRMS of 0. All the comparisons were made after transforming back the predicted values to the original form and scale using the inverse transformations (for example calculating the exponential of values in logs). We show the absolute percentage prediction error as well, to evaluate the relative error size.

4.2 The Models

The SARIMA models were built according to the methodology described in [4], with an iterative model selection procedure mainly based on the analysis of the autocorrelations. We used the time series module of statistical software package SPSS [19]. The selected final models were: SARIMA (100)(100), using raw values; SARIMA (100)(100), in logs; SARIMA(200)(010), logs + seasonal differences; SARIMA(110)(210), logs + first + seasonal differences. To separate as much as possible the effects of the preprocessing stage(s) from the effects of the model identification procedure, we did not select the best network and the best parameters in each case: this important issue, recently addressed e.g. by [1] or [17], will be the subject of further research. We selected a simple neural network architecture, using the same set of training parameters along the whole evaluation procedure. The functional specification used for the neural network models was: $y_t = f(y_{t-1}, y_{t-2}, ..., y_{t-12})$. When comparing with the SARIMA(100)(100), which has a lag at $t - 13$, we included y_{t-13} among the explanatory variables. We intentionally used a simple specification limiting the number of lags used in the last model, comparing a neural network based on the first 12 lags with the SARIMA (110)(210) which includes the lags 1,12,13,24 and 25 (refer to [4], chapter 9).

The network was a single hidden layer backpropagation network, with full interconnections, built using 'Previa', developed and kindly made available by Elseware S.A.[7] which the authors gratefully acknowledge. No direct connections between input and output units were present. The topologies were the following: 13-5-1 hidden units, using raw values, lags 1-13; 13-5-1 hidden units, data in logs, lags 1-13; 12-5-1 hidden units, logs + seasonal differences, lags 1-12; 12-5-1 hidden units, logs + first + seasonal differences, lags 1-12. The input and target vectors were normalized within the range [-.8,+.8]. The input activation functions were linear, the activation functions in the hidden units were sigmoids. The learning algorithm used is backpropagation, with learning rate 0.9 and momentum 0.1, and the weights were initialized randomly within the range [-0.1,0.1],

Seasonal ARIMA and Neural network - 30-steps out of sample predictions					
Type of preprocessing	*ADF - DHF tests*	*NRMS SARIMA*	*NRMS neural net*	*Average abs. % error SARIMA*	*Average abs.% error neural net*
raw values, (100) (100)	A 5% - A 5%	0.749	0.504	34.80%	22.64%
logs, (100)(100)	A 5% - A 5%	0.930	0.560	40.30%	24.50%
logs+sdiffs, (200)(010)	A 10% - R 5%	0.568	0.469	27.08%	26.42%
logs+first diffs+sdiffs, (110)(210)	R 5% - R 5%	0.633	> 1	29.88%	97.82%

Figure 2: Neural network and seasonal ARIMA out of sample predictions with different preprocessing methods. The error is computed after converting back the predictions to the original form. In the unit root tests column, 'A' stands for accepted, 'R' for rejected, followed by the confidence value.

with a uniform distribution. The regularization was based on weight decay, with a rate of $5 * 10^{-5}$, chosen by splitting the in-sample data set into a training (80%, 85/01-92/12) and test set (20%, 93/12-94/12), and using the test set to check the generalization performance with different weight decay rates; after choosing the appropriate weight decay rate, the whole available in-sample data set was used for training. The training was stopped when the error on the training set reached a minimum. After the choice of the decay rate, the network was erased and the training restarted. The final splitting of the available data set was done as follows: training (85/01-94/12; (120)), test (none), out-of-sample multi-step (95/01-97/06; (30)).

4.3 Results

We evaluated the predictions of the 8 estimators on the postprocessed data, i.e. converted the predicted values back to the original form. Results and pictures of the winner models are depicted in Figures 2 and 1. The neural networks are always more accurate than the SARIMA models, with the exception of the double difference model. Concerning the impact of the preprocessing techniques, the two performance indexes show somehow contradicting outcomes: the lowest absolute percentage error is obtained by the neural network using the raw values, where the lowest root mean squared error is associated with the log seasonal differences. The graphs in Figure 1 may help us to understand the reason: the predictions based on the raw values (on the left side) totally miss to predict the new forthcoming upward trend; the multi-step forecast of the neural estimator is mainly resembling the average level of the immediate past, with a partial reconstruction of the seasonal pattern which makes it better than a simple average (NRMS < 1). On the right side, the neural forecasts, based on the log seasonal differences, have a very high variance and they swing very much around the trend line, producing high absolute percentage errors, but nevertheless they can capture the new trend; moreover, the high variance of the estimator resembles the high variance of the target data set: the resulting NRMS error is quite low. It is interesting to compare the 'true' trendline, obtained with a 5 degrees polynomial interpolation of the actual values, with the estimated trend line, based on the predicted values: they are quite close, even if the prediction errors are still high.

5 Conclusions

Economic time series are rarely stationary and normally distibuted. Locating and testing stationarity is not trivial, as testified by the existing literature about unit-root tests in linear modelling (see for example [2] chapter 4). We have outlined that there are a number of reasons for identifying nonstationarity and trends, using appropriate preprocessing techniques: data representation can help to "prewire" in the forecasting model information about the underlying structure (like trends and seasonal patterns) and to obtain better forecasts. Our experiment based on the mortgage loans time series showed that the prediction performance of a simple neural network model can be considerably improved by an appropriate preprocessing stage (log seasonal differences) which addresses stationarity, seasonality and corrects the time series distribution.

The resulting estimator was capable of forecasting a trend which would be missed without the appropriate preprocessing, but it showed a relatively high average error. The question is whether a high variance estimator which can capture the tendency line or a more conservative, lower variance estimator with a lower absolute error which misses the new trend should be preferable. The answer to this question clearly depends on the purpose of the predictions; obviously the best choice would be a low variance, low bias estimator: a causal model, introducing the effect of external factor could be taken into consideration. Obviously, more research should be devoted to better understand why and how the choice of preprocessing can influence neural networks' prediction performances, and possibly to develop new tests and methodologies to choose the right filters and/or transformations.

References

[1] Anders U. and Korn O., Model Selection in Neural Networks, Neural Networks, 12, 309-323, 1999.

[2] Banerjee A., Dolado J., Galbraith J. and Hendry D., Cointegration, Error Correction and the Econometric Analysis of Non-stationary Data, Oxford University Press, New York, 1993.

[3] Bierens H.J., Testing the Unit Root with Drift Hypothesis Against Nonlinear Trend Stationarity, with an Application to the US Price Level and Interest Rate, Journal of Econometrics, 81, 29-64, 1997.

[4] Box G.E.P., Jenkins G.M. and Reinsel G.C., Time Series Analysis, Forecasting and Control, Prentice Hall, 1994.

[5] Bishop C.M., Neural Networks for Pattern Recognition, Oxford University Press, New York, 1996.

[6] Dickey D.A., Hasza D.P. and Fuller W.A., Testing for Unit Roots in Seasonal Time Series, Journal of the American Statistical Association, 79, 355-367, 1984.

[7] Elseware S.A., Previa Users' Manual, Elseware S.A., 1997.

[8] Franses P.H., Time Series Models for Business and Economic Forecasting, Cambridge University Press, Cambridge (UK), 1998.

[9] Freisleben B. and Ripper K, Economic Forecasting Using Neural Networks, in Proceedings of the 1995 IEEE International Conference on Neural Networks, 2, 833-838, 1995.

[10] Geman S., Bienenstock E. and Doursat R., Neural Networks and the Bias/Variance Dilemma, Neural Computation, 4,1-58, 1992.

[11] Hatanaka, M., Time-series Based Econometrics, Oxford University Press, NY, 1996.

[12] Granger C. and Tersvirta T., Modelling Nonlinear Economic Relationships, Oxford Univ. Press, 1993.

[13] Moody J., Economic Forecasting: Challenges and Neural Network Solutions, Keynote Talk presented at the International Symposium on Artificial Neural Networks, Hsinchu, Taiwan, December 1995, ftp://neural.cse.ogi.edu/pub/neural/papers.

[14] Nelson C. and Plosser C., Trends and Random Walks in the Macroeconomic Time Series, Some Evidence and Implications, Journal of Monetary Economics, 10, 139-162, 1982.

[15] Perron P., The Great Crash, the Oil Price Shock and the Unit Root Hypothesis, Econometrica, 57, 1361-1402, 1989.

[16] Refenes A.P.N. (editor), Neural Networks in the Capital Markets, John Wiley & Sons, 1995.

[17] Refenes A.P.N., Zapranis A.D and Utans J.:, Neural Model Identification, Variable Selection and Model Adequacy, in Decision Technologies for Financial Engineering, Proceedings of NNCM 96, Editors: Weigend A., Mostafa Y.A. and Refenes A.P.N., pp. 243-261, World Scientific Publishing, Singapore, 1997.

[18] SAS Institute inc.,SAS/ETS User's Guide ver. 6.12, SAS Institute inc., 1996.

[19] SPSS inc., SPSS Trends 8.0, SPSS inc., 1998.

[20] Virili F., Freisleben B., Preprocessing Seasonal Time Series for Improving Neural Network Predictions, in H. Bothe, E. Oja, E. Massad, C. Haefke (eds.), Proceedings of CIMA 99: Computational Intelligence - Methods and Applications, Rochester (NY), pp. 622-628, 1999.

[21] Weigend A.S. and Gershenfeld N.A. (editors), Time Series Prediction: Forecasting the Future and Understanding the Past, Addison-Wesley, 1994.

[22] Weigend A. Huberman B.A. and Rumelhart D.E., Predicting Sunspots and Exchange Rates with Connectionist Networks, in Nonlinear Modelling and Forecasting, Editors: Casdagli M. and Eubank S, Addison-Wesley, Redwood City (CA), 1992.

[23] White H., Economic Prediction Using Neural Networks: the Case of IBM Daily Stock Returns, in Proceedings of the 1988 IEEE International Conference on Neural Networks, 2,451-58,1988.