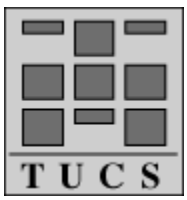
A Neural Network Model for Prediction: Architecture and Training Analysis

Iulian Nastac

Turku Centre for Computer Science,
Institute for Advanced Management Systems Research,
Lemminkäisenkatu 14, FIN-20520 Turku, Finland
e-mail: Iulian.Nastac@abo.fi

Eija Koskivaara

Turku School of Economics and Business Administration,
Information Systems Science,
Lemminkäisenkatu 14, FIN-20520 Turku, Finland
e-mail: Eija.Koskivaara@tukkk.fi



Turku Centre for Computer Science
TUCS Technical Report No 521
April 2003
ISBN 952-12-1150-4
ISSN 1239-1891

Abstract

The main purpose of the present paper is to establish an optimum feedforward neural architecture and a well suited training algorithm for financial forecasting. The artificial neural networks (ANNs) ability to extract significant information provides valuable framework for the representation of relationships present in financial data. The evaluation of the computing effort involved in the ANN training shows us that a good choice for the network architecture and training algorithms can substantially improve the achieved results.

Keywords: artificial neural network (ANN), financial forecasting, training algorithm, number of training cycles

Data Mining and Knowledge Management Laboratory

1. Introduction

Financial forecasting is a very difficult task due to the lack of an accurate, convincing model of the economy. Constructing reliable time series models is challenging due to short data series, nonstationarities, and nonlinear effects. Extensive research has been conducted about the application of artificial intelligence (AI) to financial forecasting in today's globalized trading environment. With the advancements being made in computer and telecommunication technologies today, financial markets are becoming more and more interrelated and fundamental factors will become increasingly critical to financial analysis. Several approaches have been proposed for time-series prediction, based on artificial neural networks (ANN) models [1] [2] [4].

In this paper, we present an ANN forecasting model which is very useful in the case where there is a huge amount of data that imply the presence of correlations across time. We also survey some training algorithms in order to establish which of them are the most suitable for our financial prediction tool.

2. Forecasting model

A good choice of the training data set is not a trivial matter when one wants to make a good prediction. Data preprocessing and data selection remain essential steps in the knowledge discovery process for real world applications and greatly improve the network's ability to capture valuable information when correctly carried out [5].

In Fig. 1 we present the basic idea in training a feedfoward ANN to be a predictor. Assume that the lines across the top of the figure carry data whose values we wish to forecast.

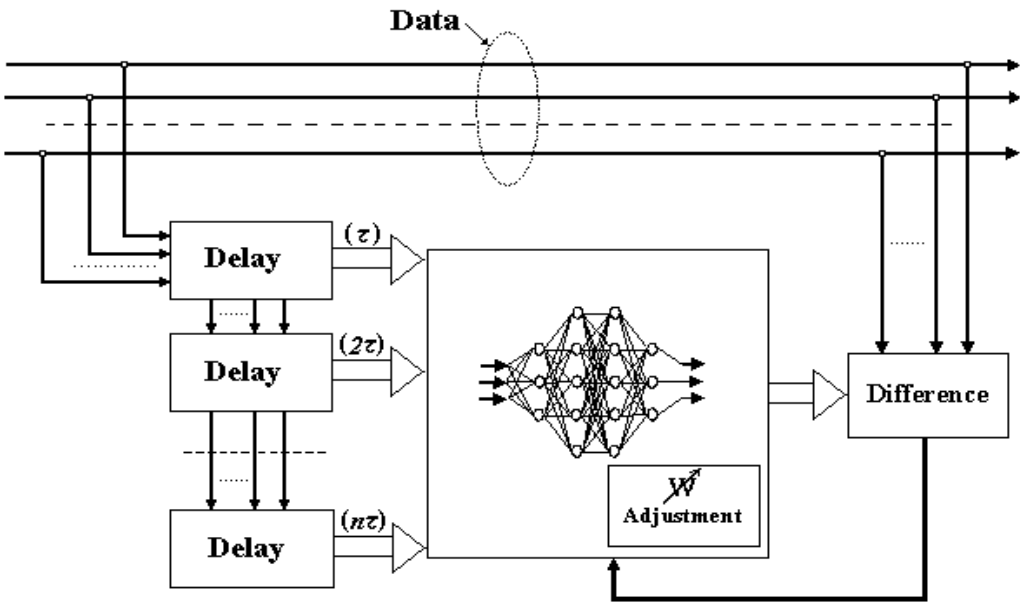


Fig.1. Forecasting Neural Network Model

Delay coordinates of the data are used to construct representations of the current states. For learning purposes, the network inputs consist of delayed values of the data, and the target outputs consist of the current values of the data. Therefore, the network tries to match the current values by adjusting a function of their past values. After training, when the input to the network bypasses the first delay unit, the output of the network is a prediction of the values the data will have in the future. Assume that each delay unit delays its input for τ time steps and that the network requires a negligible amount of time to compute its output from its input. Then the trained network provides estimates for the values of the data τ steps in the future.

At every moment in time, data lines provide a vector $x_t(p_1, p_2, \ldots, p_m)$, where m is the number of data lines, and each parameter p_i is associated with a specific financial parameter that has its own evolution in time. In this way the neural network is able to use the possible correlations that exist among the data lines. The state at time t of a variable x, denoted $S(x_t)$ is constructed from the observed values:

$$x_t, x_{t-t}, x_{t-2t}, \ldots, x_{t-(d-1)t}$$

The parameter d is considered the *window size*, the number of most recent historical values that will be used to construct a state. It is assumed that a functional relationship between the current state and the future state exists:

$$x_{t+t} = f(S(x_t))$$

The ANN that we use is a predictor approximating the function f.

An improved trading strategy uses forecasts further into the future than next-step. While the current architecture at time t produces a forecast for $t+t$, an extension of length $k>1$ refers to a forecast for $t+kt$. There were two different methods by which the existing algorithm could be changed to produce such extended forecasts.

The first method is to modify the training phase to take into account the required extension. For an extension of length k, the evolution of the state, for time $t+kt$ in the training set, is now x_{t+kt} .

The alternative is to leave the existing algorithm unchanged, to construct a next-step forecast and to treat the forecast as an observed value to produce another next-step forecast for further ahead. Using this iterative process, a forecast can be extended as many steps as required, but in this case, each step increases the forecasting error. For this reason we prefer the first method. The length kt of any forecast will depend to a large extent on the degree of accuracy of the prediction.

In order to properly evaluate the proposed procedure, we perform a number of benchmark comparisons of four most efficient training algorithms: Scale Conjugate Gradient (SCG), Levenberg-Marquardt (LM), Resilient Backpropagation (RP) and Conjugate Gradient with Powel-Beale Restarts (PBR). This way, we try to find the better algorithm that fits our model.

As the training techniques are fairly similar for the ANNs with an arbitrary number of inputs and outputs, it implies that the achieved results will be true even in the general case of a highly dimensional ANN.

3. Choice of sample and data

For the study, we have selected the data regarding the Health Care Services from one big municipality provided by internal auditors. We have collected seven years' (1995-2001) monthly account values from these units. The year 2001 was selected as the testing period and all the previous years were used for training the ANNs. In the raw data there were hundreds of accounts per unit. In this preliminary experiment we have not used all accounts. We selected those accounts per unit that were most stable compared to previous year's values. These accounts were the largest and the most significant ones like salaries and salary-related accounts.

4. Results and discussion

The network topology used is a feedforward ANN architecture with one hidden layer. As transfer functions we used hyperbolic tangent sigmoid functions for the hidden layer and linear functions for the output layer.

We decided to have about four accounts per model in order to keep it user-friendly. We improved the model from Fig. 1 considering an additional kind of data (as we can see in Fig. 2) which indicates the period of time, named the month indicator in our case where the data presented a yearly cycle trend. Then the number of data lines became five.

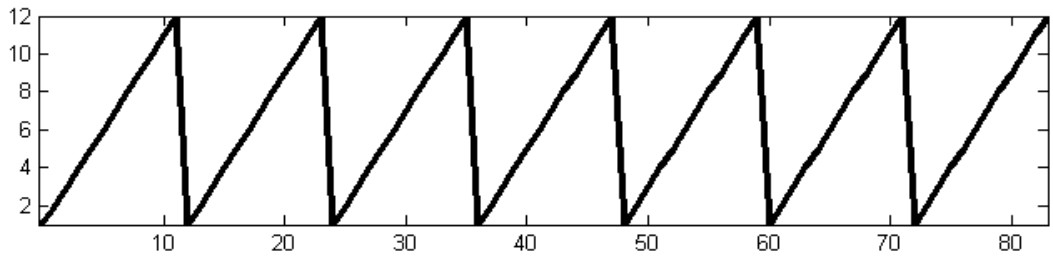


Fig. 2. The month indicator.

On the other hand we wanted our network coefficients to be overdetermined. Generally it is recommended to have at least two times more training samples than the number of the weights. Our goal, then, is to achieve a fit which will not only match the training data, but will also work well when new data are provided to the network. Poor performance after training is an indication of *overfitting*. Generally the solution to *overfitting* is a reduction in the number of hidden units or even in the number of inputs. Our number of samples is limited to 84. Therefore we decided to use four accounts (plus month indicator) and one or two delay units. The ANN architecture for one delay unit (1D) was 5-10-5 and the structure for two delay units (2D) was 10-15-5. We performed several tests in order to determine suitable parameters and topologies for every model. The data in our model were pre-processed by scaling inputs and targets so

that they always fall within the range [-1, 1]. All the above conditions do not restrain the generality of the method, instead they try to comply with the limited computing resources available and also help the understanding by providing intuitive results.

The data used (from 1995 to 2001) are organized into a matrix with 5 rows and 84 columns. The year 2001 was selected as the testing period and all the previous years were used for training and validation the ANN. For each learning algorithm we performed two procedures: training without validation and training with validation. In the first case we used as training set all the data from 1995 to 2000 and in the second one we extracted in random mode some values from this interval in order to use them as validation set [3]. The achieved results during the training phase are presented in the next table. Each value that represents the number of training cycles is a round average over twenty different training experiments.

Training algorithm	**SCG**		**LM**		**RB**		**PBR**	
	1D	**2D**	**1D**	**2D**	**1D**	**2D**	**1D**	**2D**
without validation	790	201	43	29	3767	932	890	213
with validation	114	75	21	15	198	87	156	78

Tab. 1. Values of training cycles number for varied training algorithms

At the first sight we state that the Levenberg-Marquardt (LM) algorithm is the best in this comparison. But LM algorithm always computes the approximate Hessian matrix, which is a complex task. If the network is very large, then we may run out of memory. In this case the right choice is the Scale Conjugate Gradient (SCG) algorithm.

Concerning the output results, after each training session, we observed a homogeneity of variances of the difference between the neural network outputs and desired outputs. There is no reason to compare the accuracy of the results among the mentioned algorithms. The only parameter that can establish which algorithm is more suitable remains the convergence speed (the number of training cycles).

In Fig. 3 we show an example (parameter no. 3 which is a salary account) of the dynamics the ANN model has learnt in 2000 and what it is able to provide for 2001. The presented graphs were obtained after training with SCG algorithm.

Remember that the year 2000 was included in the training set and 2001 in the test set. This is the reason why in 2001 the ANN has not very good performance especially because of overfitting. We mark with thick line the real values and with thinner line the values produced by neural network. The ANN values are within the +/-10% interval (limited by the dot lines) of the real values especially when we use training with validation stop, which is the best solution for a limited set of data. However, the ANN model also predicts the monthly variation of the salary expenses. For accounts that have a trend or causality this kind of model adds values enabling us to allocate money into the right places at the right time.

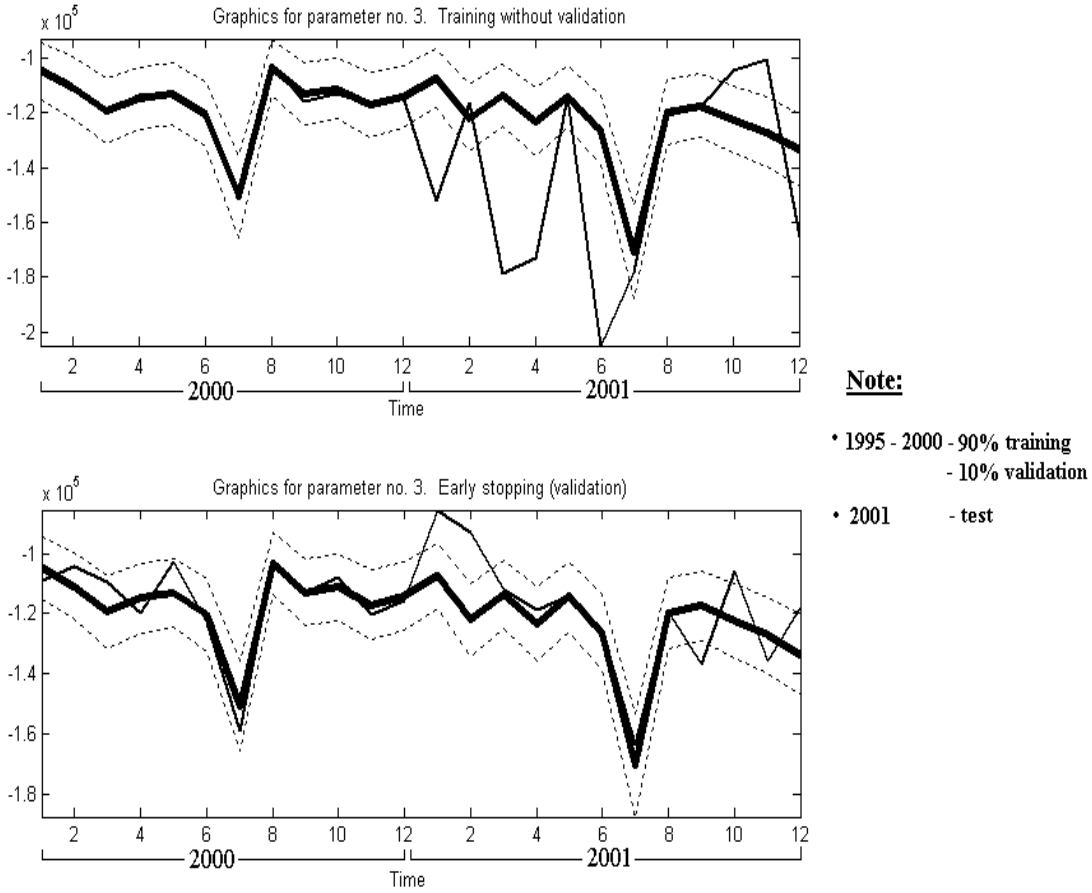


Fig. 3. Real data (thick line) and neural network values (thinner line)

Concerning the prediction further into the future we have obtained satisfactory results with extensions of length k=2 and k=3. The accuracy of forecasts decreases slowly as long as forecasts are made further ahead in time.

5. Conclusions

This paper has described a basic neural network engine for financial prediction. The ANN-model was trained and tested with the monthly account values of a big organization. It has shown how an ANN can be able to perform complex and fast rule forecasting, using LM or SCG training algorithms.

The power of the ANN model described in this study lies in its ability to model the monthly dynamics of account values. The ANN model works best with accounts with a trend and with some causality to each other. These kinds of accounts in our models are salaries and salary-related accounts. The ANN model does not work very well with those accounts which are mainly based on occasional decisions.

In the studied cases we found that the optimum number of delay units is two. An increased number grew up the architecture of neural networks and did not have a significant influence over the accuracy of the result.

A small change in the initial project (e.g. some re-evaluation of the ANN performance at time point when a certain amount of experience has been accumulated) might require the repetition of the entire training phase, including all the shortcomings deriving from that, i.e. a long processing time, the possible occurrence of an undesired local minimum of the performance function, etc. We can imagine that the training process for a case with 100 data lines requires a large number of processing cycles, which can occasionally reach and even outnumber hundreds of thousands. In order to overcome this disadvantage our research is being conducted to use the retraining procedure [6]. In this way we can re-train a viable ANN structure at a certain moment of time such that it will support variations of the initial input-output function. Smaller values of the training cycles give the prediction results faster, which is an important goal.

References

[1] Bashir, Z., El-Hawary, M.E., Short term load forecasting by using wavelet neural networks, *Proc. of Canadian Conference on Electrical and Computer Engineering (CCECE*), pp. 163-166, 2000.

[2] Coakley, J.R., Brown, C.E., Artificial Neural Networks in Accounting and Finance: Modeling Issues, *International Journal of Intelligent Systems in Accounting, Finance & Management*, 9, pp. 119-144, 2000.

[3] Hagan, M.T., Demuth, H.B. and Beale, M., *Neural Networks Design*. MA: PWS Publishing. Boston, 1996.

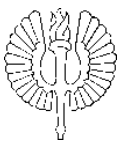
[4] Mosmans, A., Praet, J-C, and Dumont, C., A decision support system for the budgeting of the Belgian health care system. *European Journal of Operational Research*, June 1, pp. 449-60, 2002.

[5] Nastac, I., Contributions in Technical Systems Quality Modelling through the Artificial Intelligence Methods. Ph.D. dissertation, Polytechnic University of Bucharest, 2000.

[6] Nastac, I., and Matei, R., A Retraining Improvement of Feedforward Neural Networks, *TUCS Technical Report* 504, Turku Centre for Computer Science, February 2003.

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

http://www.tucs.fi/



University of Turku
- Department of Information Technology
- Department of Mathematics

Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

Turku School of Economics and Business Administration
- Institute of Information Systems Science