INFORMATICA, 1999, Vol. 10, No. 2, 231–244

Comparison of ARMA and Multilayer Perceptron Based Methods for Economic Time Series Forecasting

Aistis RAUDYS, Jonas MOCKUS
Institute of Mathematics and Informatics
Akademijos 4, 2600 Vilnius, Lithuania
e-mail: giena@takas.lt

Abstract. In this paper two popular time series prediction methods – the Auto Regression Moving Average (ARMA) and the multilayer perceptron (MLP) – are compared while forecasting seven real world economical time series. It is shown that the prediction accuracy of both methods is poor in ill-structured problems. In the well-structured cases, when prediction accuracy is high, the MLP predicts better providing lower mean prediction error.

Key words: ARMA model, multilayer perceptron, comparison, prediction, forecast, economic time series.

1. Introduction

Forecasting the economical times series is an important task in modern data analysis. The problem can be formulated as follows. Let $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_R$ be a multivariate time series consisting of R p-variety vectors. An objective is to forecast the first component of a vector

$$\boldsymbol{X}_j = \begin{bmatrix} X_{1j} \\ X_{2j} \\ \ldots \\ X_{pj} \end{bmatrix}$$

on a basis of information contained in a sequence of p-variety vectors $X_{j-1}, X_{j-2}, \ldots X_{j-K}$, where K is the number of vectors in a past we want to use. Usually for prediction of each particular value X_j we use only a finite number (K) of "history" vectors $X_{j-1}, X_{j-2}, \ldots, X_{j-K}$, where $K << R$. For prediction of parameter $Y_j = X_{1j}$ one seeks for a functional relationship $Y_j = f(X_{j-1}, X_{j-2}, \ldots, X_{j-K})$.

There is a number of methods for solving time series forecasting problem. Most popular are Regression Trees, Neural Networks, the Auto Regression and Memory Based Reasoning. We concentrate on two of them: the Auto Regression Moving Average (ARMA) model and the Multilayer Perseptron (MLP).

In the **ARMA prediction model**, it is supposed the vectors $X_1, X_2, \ldots, X_R$ are samples of a stationary Gaussian process (see, e.g., Box and Jenkins, 1974), and

$$Y_t^* = \sum_{l=1}^{K} \sum_{i=1}^{p} a_{p(l-1)+i} X_{it-l} + \sum_{i=1}^{q} b_i \varepsilon_{t-i} + \varepsilon_t. \tag{1}$$

We assume that

$$X_{it-l} = 0, \quad \text{if} \quad t \leqslant l,$$

and

$$\varepsilon_{t-i} = 0, \quad \text{if} \quad t \leqslant i,$$

and where Y_t^* is tth day (a moment of the time series) prediction, X_i is ith day vector, ε_t is tth day prediction error, K is a number of "history" moments, p is vector size, q is number of Moving Average (MA) parameters.

The training consists in finding unknown coefficients a_i, b_i from a "learning-set" data.

The multilayer perceptron calculates an output as follows

$$Y_t^* = \phi_{output}\Big(\sum_{j=1}^{h} v_j \phi_{hidden}\Big(\sum_{l=1}^{K} \sum_{i=1}^{p} w_{p(l-1)+i} X_{it-l} + w_0 \Big) \Big), \tag{2}$$

where $\phi(x)$ is a non-linear activation function, e.g. $\phi(x) = \dfrac{1}{1+e^{-x}}$, Y_t^* is tth day (a moment of the time series) prediction, w_i is MLP hidden layer coefficients, v_j is MLP output layer coefficients, p is vector size (dimensionality), h is number of hidden neurones, K is number of "history" moments.

The prediction task is to find the coefficients w_i, v_j.

An objective of the paper is to apply both the ANN and the ARMA prediction methods to several real data sets and to compare their accuracy. In the next section, the details about the methods are presented. In the Section 3, the economical data sets used in the experiments are given, in the Section 4 the criteria to compare accuracy of different prediction methods are discussed. The Section 5 contains the results of experimental analysis.

2. Prediction Methods

2.1. *ARMA*

We define residuals in prediction by recurrent expressions:

$$\varepsilon_t = Y_t^* - Y_t, \tag{3}$$

where Y_t^* are forecasted values, Y_t are real values.

In order to find unknown coefficients of the ARMA model the following sum of squares is minimized

$$f(x) = \log f_m(x), \quad f_m(x) = \sum_{t=1}^{R} \varepsilon_t^2, \tag{4}$$

where ε_t is the prediction error of the day t, R is the number of moments.

An advantage of residual minimisation is that one may see directly how the objective depends on unknown parameters The logarithm of the sum is often used to decrease the objective variation by improving the scales (see, e.g., Mockus, 1997).

2.2. *Multilayer Perceptron*

The MLP consists of several layers of neurons. The neurons are connected with each other as shown in Fig. 1.

The input signals are submitted to hidden layer neurons. One can use several hidden layers where the outputs of each layer are submitted to inputs of the next layer neurons. In terms of the standard regression models, each single neuron uses a non-linear activation function:

$$Y_t^* = \phi\Big(\sum_{j=1}^{h} w_j X_{it} + w_0\Big), \tag{5}$$

where Y_t^* is the output of the neuron (the forecasted value), X_{it} are inputs of the neuron (the values used for forecasting), w_i are the neuron's parameters (weights).

The idea of the ANN model is that the activation function roughly represents the activation of a real neuron. Outputs of the highest hidden layer neurons are submitted to

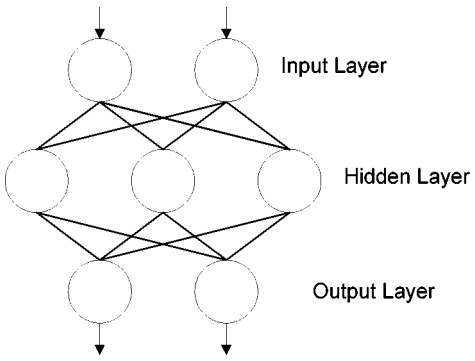

Fig. 1. Model of MLP.

the output layer neurons. In our analysis, only one hidden layer perceptron was used with following data processing function:

$$Y_t^* = \phi_{output}\Big(\sum_{j=1}^{h} v_j \phi_{hidden}\Big(\sum_{l=1}^{K}\sum_{i=1}^{p} w_{p(l-1)+i} X_{it-l} + w_0\Big) + v_0\Big), \tag{6}$$

where Y_t^* is the prediction of the t-th day, X_i is the data of the day i (i-th day vector), K is the "history length", p is the dimensionality of input vector, w_i are MLP hidden layer coefficients, v_j are MLP output layer coefficients, N – is a number of inputs, h is the number of neurons in the hidden layer, $\phi(x)$ are the activation functions. In the output layer

$$\phi_{output}(x) = c\Big(\frac{1}{1+e^{-x}} - 0.5\Big), \tag{7}$$

where c is a scaling parameter to fit the MLP outputs to the data.

In the hidden layer we used a traditional sigmoid function:

$$\phi_{hidden}(x) = \frac{1}{1+e^{-x}}.$$

The non-linearity of activation function $\phi(x)$ means that the perceptron is a non-linear forecasting algorithm. We use MLP to forecast Y_t on a basis of an information contained in K preceding vectors $X_{i\ t-l}(i = 1, \ldots, p,\ j = 1, \ldots, k)$.

Typically in order to find unknown coefficients $w_j,\ j = 0, \ldots, K \times p$ and $v_i,\ i = 0, \ldots, h$ we are minimising a sum of squares cost function:

$$f_m(x) = \sum_{t=1}^{R} (Y_t^* - Y_t)^2, \tag{8}$$

where Y_t is a real value from the learning-set data (a desired output, a target), Y_t^* is a forecasted value, R is a size of a learning-set.

The cost function $f_m(x)$ depends on $K \times p + h + 2$ unknown parameters represented as vectors ($w_i;\ i = 0, \ldots, K \times p$ and $v_j;\ j = 0, \ldots, h$). Expression (8) shows that the residuals Y_j are non-linear functions of parameters w_i if the activation function $\phi(x)$ is non-linear. The minimum conditions

$$\frac{\partial f_m(x)}{\partial w_i} = 0, \quad i = 0, \ldots, K \times p, \quad \text{and} \quad \frac{\partial f_m(x)}{\partial v_i} = 0, \quad i = 0, \ldots, h \tag{9}$$

result a system of non-linear equations. As usual this system has multiple solutions. To find the perceptron's weights an iterative training procedure is used. Most often a gradient descent minimization is applied

$$\boldsymbol{w}_{(t+1)} = \boldsymbol{w}_{(t)} - \eta \frac{\cos t_l}{w}, \tag{10}$$

where $\boldsymbol{w}_{(t)}$ is a vector composed from all component weights at iteration t, η is called a learning step. A popular alternative optimization method is the genetic algorithm.

An important parameter in iterative MLP training is the learning step η. Typically, for each new data set it is chosen after few preliminary experiments (see. e.g., D.E. Rumelhart *et. al.*, 1986), in some computer implementations (MATLAB, for example) parameter η increases if the cost function $f_m(x)$ decreases, and is reduced if $f_m(x)$ begins to increase.

The MLP training process encounters a multi-modality problem, similarly as in the case of ARMA model. Thus, the perceptron's starting weights are important. Traditionally while training the perceptron the starting weight vector is selected randomly. We initialized the weights randomly, and, as a competing approach, we used our "active initialisation" technique (Raudys, 1998). Here we mapped the data into a two-variety space, and in this space, i controlled the hidden layer weights interactively. Then gradually, step by step, we returned to the original multivariate feature space. For the real world data considered in this paper both initialisation approaches resulted the same accuracy. Therefore, in tables below we present the results obtained for the traditional random weights initialisation.

3. Real World Data Sets Used in Experiments

To compare the accuracy of both prediction methods we used several sets of real world economical data obtained from different sources. Define a data set as a table of real numbers

$$X = \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1n} \\ X_{21} & X_{22} & \dots & X_{2n} \\ \dots & \dots & \dots & \dots \\ X_{p1} & X_{p2} & \dots & X_{pn} \end{bmatrix}. \tag{11}$$

Here each column represents one event in the time series data

$$\boldsymbol{X}_j = \begin{bmatrix} X_{1j} \\ X_{2j} \\ \dots \\ X_{pj} \end{bmatrix}. \tag{12}$$

The ith row we call as the ith feature:

$$[X_{i1} X_{i2} \dots X_{in}]$$

We predict values of only one (the main) feature for the next day.

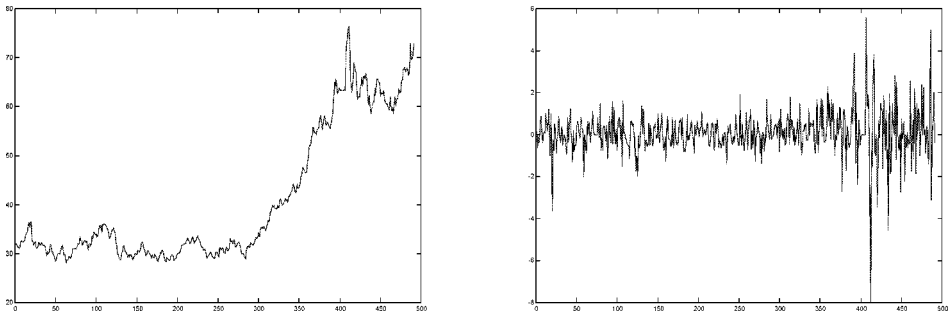

Fig. 2. Intel data set 1-st feature (original on the left, transformed on the right).

Seven real data sets were used to compare prediction algorithms. On some data sets, unsatisfactory results were obtained. In an attempt to improve the prediction accuracy, in addition to the original data, we transformed outputs in a following way

$$\begin{aligned} X_i^{tr} &= X_i - X_{i-1}, \quad i = 2, \ldots, R, \\ Y_i^{tr} &= Y_i - Y_{i-1}, \end{aligned} \tag{13}$$

where X_i^{tr} is the transformed data, X_i is the original data, Y_i^{tr} is transformed feature we are trying to forecast, Y_i^{*tr} is forecasted transformed feature, R is a number of records in the original old data.

Transformed vector is a difference between today ant yesterday values. Ill Structured Data: there is no apparent structure of data, which can be utilized for the prediction.

3.1. *London Stock Exchange Data*

Data set name is `stock`; number of features (dimensionality of the time series) is 23; number of records is 1383. Data represent London Stock Exchange closing rates in the period of approximately three years. Data contains an information such as currency exchange rates, Daw Jones indexes, closing rates of stocks of some biggest companies, etc. This data set is the largest one in this research. Therefore, in order to reduce an influence of the high data dimensionality we performed a feature selection. Thus, our experiments have been performed with 3 data sets containing 2, 8 and all the 23 features. Data was obtained from Prof. A. Long from City University, London, UK.

3.2. *AT&T Share Data*

Data set name is `att`; dimensionality is 1; number of records is 488; Data contains AT&T stock closing rates. Data was obtained from A.S. Soofi.

3.3. *Intel Stock Data*

Data set name is `intel`; dimensionality is 1; number of records is 492. Data contains closing rates of Intel stocks (see Fig. 2). Data was obtained from A.S. Soofi.

3.4. *Call Centre Data*

The data set name is `call`, the number of features is 8. The fifth column of data set `call` represents the call rate of some call center during one year (365 days). The call rate is the main feature in this case. The rest columns show the "external" features used to forecast the call rate. The external features include the date and a number of indicators of special events, such as the day of mailing new information to customers, very hot or very cold day, important sporting event etc. Data contains number of calls per day. Data was obtained from Vidas Plačiakis.

3.5. *Lithuanian Banks Stock Data*

Data set name is `bank`; number of features is 8 (stock session code, and stock rates of 7 major Lithuanian banks), a number of records is 120. Data was obtained from J. Juodagalvytė, Vytautas Magnus University, Kaunas, Lithuania

3.6. *Currency Exchange Data*

Data set name is `currency`; a number of features is 4 (exchange rates to US $ of four major world currencies – DM, FFr, UK pounds, Yen), a number of records was 464.

4. Methodology of Experiments

To evaluate the forecasting accuracy **the data sets where split into three equal parts**. The first part – *a learning-set* – was used for training-estimation of unknown prediction equation parameters (the perceptron weights, ARMA parameters). The second part – *a validation-set* – was used to select the best variant of the prediction methods: the number of inputs, the number of hidden neurones, an optimal stopping moment in MLP training, the ARMA model order parameters, etc. The third part of each data set was used to test performances of final variants of MLP and ARMA models. It is called *a test-set*.

ARMA software used in experiments was self-optimising (see Mockus, 1997), and the learning and validation sets were used together to estimate both the prediction model architecture and the model parameters. The MLP software was not self-optimising. Therefore we applied the MLP predictor several times sequentially changing the number of inputs (a number of previous observation vectors used for forecasting) and also the number of neurons in the hidden layer. In order to have a zero mean and unit variance of all features, the data was normalized. The learning rate η was set to 0.1, The MLP was trained on the learning-set. The best set of parameters was determined using the information in the validation-set.

For evaluation of the average **accuracy of both prediction methods two criteria were used**. First one is *the relative efficiency of the method in comparison with the "random walk"* (random walk predicts: tomorrow-will-be-as-today) method:

$$\vartheta = \frac{m_{forecasted} - m_{rw}}{m_{rw}} 100, \tag{14}$$

where $m = \sqrt{\dfrac{\sum_{t=1}^{R}(Y_t - Y_t^*)^2}{R}}$, $Y_0 = 0$, $m_{forecasted}$ is a mean square prediction error (MSPE), m_{rw} is MSPE of the random walk method, Y_t^* is the forecasted feature value of the day t (the moment t), Y_t is the real values, R is the sample size.

Parameter ϑ is the efficiency coefficient of the method in comparison with the random walk: when $\vartheta = 0$ the algorithm has the same accuracy as the random walk, when $\vartheta = -100$, the algorithm forecast ideally, and when $\vartheta > 0$ the algorithm is worse than the random walk.

In addition to the relative efficiency, for evaluating the accuracy of the MLP predictor, we used *a traditional correlation coefficient* ρ *between the actual and the predicted values*

$$\rho(Y^*, Y) = \frac{\sum_{i=1}^{R}(Y_i^* - \bar{Y})(Y_i - \bar{Y}^*)}{\sqrt{\sum_{i=1}^{R}(Y_i^* - \bar{Y})^2 \sum_{i=1}^{R}(Y_i - \bar{Y}^*)^2}}, \tag{15}$$

where Y_i are the original values, Y_i^* is forecasted values, $\bar{Y}$ is mean of the original values $\bar{Y} = \dfrac{1}{p}\sum_{i=1}^{p} Y_i$, $\bar{Y}^*$ is mean of the forecasted values $\bar{Y}^* = \dfrac{1}{R}\sum_{i=1}^{R} Y_i^*$.

5. Results

The first observation which follows from our experiments is that the optimisation and the parameter estimation of the ARMA model is much faster We used several ways to present the results of the experiments. Tables contain the correlation coefficients ρ and the parameter ϑ of comparison of the prediction method with the RW values evaluated on the validation and on the test sets. In the tables, the estimates obtained on the validation set are denoted by indexes. By definition, the training set's estimates represent well only the training set. Therefore the efficiency of the prediction algorithm is evaluated by the validation and by the test-set estimates. In addition to the tables, two scatter diagrams are presented, which depict the true and the forecasted parameter values using MLP and ARMA algorithms.

In Table 1 *MS* is the a number of learning epochs (an epoch is the number of learning operations using all data set vectors in learning set). *Days deep* is the number of days back used for prediction; *# hidden* is number of neurons in the hidden layer; ρ_{train} is the correlation coefficient estimated on the learning set; ρ_{test} is the correlation coefficient estimated on the test set; ϑ_{valid}, ϑ_{train} ϑ_{test} are accuracy of the MLP predictions, in comparison to the random walk method estimated on the training, validation and the test sets (in %). The best prediction accuracy ϑ_{valid} estimated on the validation set is shown in bold.

Table 1
Results of experiments with MLP using different parameters, for the stock data set

MS	Days deep	# hidden	ρ_{train}	ρ_{test}	ϑ_{valid}	ϑ_{train}	ϑ_{test}
112	1	1	0.28023	0.283288	−58.2809	−27.5862	−31.7638
112	1	2	0.283477	0.282115	−58.2816	−27.6427	−31.765
109	1	3	0.283722	0.272426	−58.2733	−27.6777	−31.5691
111	1	4	0.285586	0.278071	−58.2859	−27.6673	−31.6706
285	2	1	0.85115	0.981302	**−87.9664**	−56.2395	−78.7807
299	2	2	0.863091	0.981265	−87.9156	−58.174	−80.5493
299	2	3	0.865426	0.979076	−87.6547	−57.1322	−77.0033
327	2	4	0.864981	0.979685	−87.731	−58.5318	−80.2028
237	3	1	0.815711	0.975249	−85.638	−53.0071	−74.9967
237	3	2	0.821114	0.973893	−85.227	−53.1312	−73.574
268	3	3	0.835287	0.970705	−84.8299	−56.4441	−78.1757
277	3	4	0.827319	0.969812	−84.6009	−55.3923	−77.0132
308	4	1	0.815684	0.976644	−83.5278	−56.0419	−84.1125
136	1	7	0.286319	0.280904	−58.2725	−27.7822	−31.71

The Table 1 reviews several estimates and shows that the efficiency of the MLP prediction depends on the number of inputs (a number of previous days used for prediction), the number of hidden neurons and, of course, on the number of learning iterations. The traditional criterion used to estimate the prediction accuracy – the correlation coefficient between the true and predicted values – often fails to evaluate the prediction performance: in some cases the MLP resulted in lower accuracy than the random walk method (see results obtained for `att, intel, currency3`), nevertheless the correlation coefficient was high. A possible reason of this phenomenon can be traced by inspecting the Fig. 3. There we see that the correlation coefficient for the MLP algorithm is rather high, however a slope of the regression line in this figure differs from the bisector which represents an ideal prediction. It means, that while solving the real life problems one needs to consider several accuracy measures. The most interesting estimates are ρ_{test} and ϑ_{test}. These two characteristics were used in Table 2 for shorter presentation of results obtained on several data sets.

In Table 2: *Data Set* is the name of the data set; *RW* ρ_{test} is correlation coefficient between Random Walk (RW) forecasted values and original values; *MLP* ρ_{test} is the minimum correlation coefficient between forecasted values and original values in the test set, during series of experiments with MLP. For MLP ϑ_{test} is the minimal value of the coefficient ϑ between the forecasted values and original values in the test set, during series of experiments with different architectures of the network and the number of training epochs. For ARMA model ϑ_{test} is ϑ coefficient between ARMA forecasted values and original values in the test set. The best prediction accuracy ϑ_{test} is in bold.

Table 2
Forecasting results for all data set using the ARMA and MLP prediction methods

Data set name	ARMA ϑ_{test}	MLP ϑ_{test}	MLP ρ_{test}	RW ρ_{test}
`att`	**0.721**	4.871	0.986	0.986
`bank(7)`	43.940	**17.244**	0.704	0.743
`stock1`	−28.436	**−28.927**	0.026	0.006
`stock2`	−54.584	**−78.780**	0.981	0.006
`stock8`	−33.005	**−86.614**	0.983	0.006
`stock23`	–	32.264	0.519	0.006
`call`	−19. 482	**−21.399**	0.665	0.535
`currency(3)`	**43.940**	89.299	0.910	0.932
`intel`	**23.273**	1356.960	0.684	0.983
`intel`$^{(\mathrm{tra})}$	−25.382	**−26.352**	0.223	0.098
`intel`*	23.273	1056.720	0.877	0.983

$^{(\mathrm{tra})}$ Transformed data set, we forecasted differences between neighbour moments (see 3, 3.3 and 6).

`stock2, stock8` – forecasting using just 2 or 8 features form all 23.

* means that MLP model is used, without non-linear activation function in output layer.

" – " in the `stock` row means that this data set was too large to be used for our software to estimate the ARMA model's parameters.

Figure 3 shows two scatter diagrams of the closing rates (on a horizontal axis) and the predicted rates (on a vertical axis). For the prediction we used only two days 2 financial parameters. Straight line in both figures characterises an exact prediction (then we have zero forecasting error). For the MPL forecast of architecture 4–3–1 the correlation coeffi-

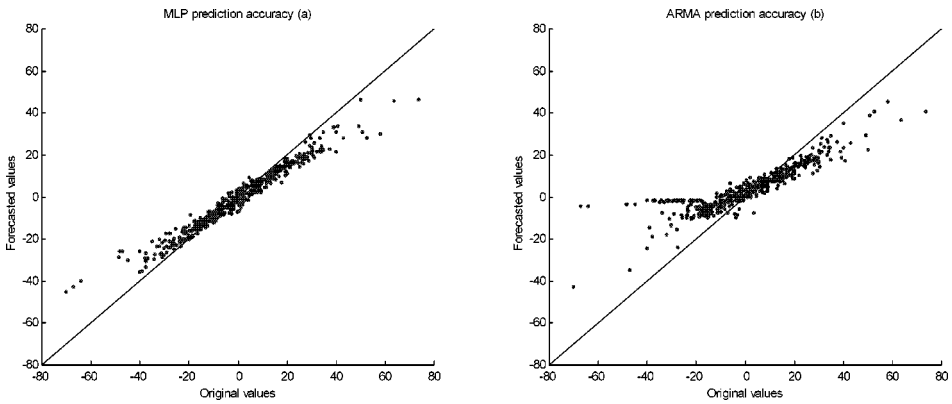

Fig. 3. London Stock Exchange closing index forecasting (the test data set) by MLP (*a*) and ARMA (*b*) methods. Points – MLP(a) or ARMA(b) prediction, plain line – ideal prediction.

cient $\rho_{test} = 0.981$, for the ARMA model the correlation coefficient $\rho_{test} = 0.898$, and for the RW prediction the correlation coefficient $\rho_{test} = 0$. We see for this data set the MLP outperforms the ARMA and RW methods in the sense of higher correlation.

The Table 2 shows that for some data sets the prediction accuracy is rather low, using both the ARMA and MLP models. Only for four data sets (stock-2, stock-8, call and intel$^{\mathrm{tra}}$) out from eleven ones we obtained higher accuracy than the trivial random walk method. For these four data sets the multilayer perceptron was more efficient. Curiously, in seven data sets, the trivial random walk outperformed more sophisticated linear ARMA and nonlinear MLP prediction methods. In such cases, the difference between the ARMA model and MLP was rather small. It suggests an absence of a statistical structure in these data sets. In situations where all participants of a financial game are making their own forecasts of variables of common interest, methods which enable to use a feedback of the variables to be predicted, the game theory, as well as additional variables with a useful information should be applied.

6. Discussion

MLP forecast can be made using several models of different architecture. The first one is when the output layer neuron acts as described by equation (7). The second does not uses the activation function at all, i.e.,

$$Y_j^* = \sum_{i=1}^{l} x_{ij}\omega_i + \omega_0. \tag{16}$$

In the first case, the predicted values are mapped into an interval (0.4, 0.6). The output of the activation function varies between 0 and 1. Thus, a largest part of the forecasted values in the learning set vary approximately in the linear part of the activation function. Then influences of very large deviations of the target – true (measured) values of the forecasted feature are reduced. Consequently, the activation function brings some stability to MLP training.

In the another case, we do not use the activation function at all, and do not map the forecasted feature. Thus, it may vary in a very wide interval. In this case, abnormal deviations of the forecasted feature values in the training set (outlyers) diminished the prediction accuracy. A reason of more successful use of the MPL predictor with the non-linear output is that use the non-linearity in the output layer of the MLP predictor makes this algorithm to be more robust to outlayers – atypical observations.

The Tables show that the ARMA model is slightly better when the data is ill-structured and thus difficult to forecast. This difference, however, is not significant since the prediction accuracy is low (lower than that of the random walk method). MLP exhibits better results than ARMA in cases when the data set is well structured and thus more suitable for prediction. On some well-structured data sets we obtained much better results comparing with the random walk. In these cases, the MLP forecasting accuracy was notably higher then ARMA one.

In the ill-structured data sets, the regular methods did not outperformed RW prediction (the highest tomorrow prediction accuracy was archived by using today values). Note that the ARMA model implements the random walk exactly if the first AR weight is equal to one and all the others are zero. That is not easy to do with the non-linear MLP. If the non-linearity is involved in the well-structured data sets then MLP may show good results.

It seems that many economical data sets are ill structured. The results are discouraging, not better than the trivial random walk. Just occasionally we outperformed RW (see `intel` $\vartheta_{test} = 23.273$, `att` $\vartheta_{test} = 0.721$ data sets). That means the structure of the currency exchange rates, stock values and similar data sets changes in time in a hardly predictable way therefore one needs additional information to make a good economical forecasting.

We obtained very low prediction accuracy for the Intel data set using both the MLP and ARMA methods. Therefore, we made an attempt to forecast first differences between values (14), and later return to the original variables. This strategy did not improved the accuracy. The forecasted (using both MLP and ARMA) differences (Y^{*tr}) was so tiny that after transformation back it seems that this values have no difference from RW forecasted values. As expected, if data in its original form is pour, it cannot be improved by forecasting the differences.

7. Conclusions

- The ARMA method is much faster than the MLP one.
- In average, the MLP model predicts well-structured data notably better as compared with the ARMA model.
- In average, the ARMA model predicts ill-structured data slightly better as the MLP.
- The MLP without non-linear activation function in the output layer is more robust to outlyers and adapts better in the case of non-stationary time series.
- The economical time series describing the highly competitive processes cannot be expected to be well-structured ones because the real results may depend on their predictions. Therefore, the additional features such as “inside information” are essential for the accurate predictions of such date.
- The average accuracy of the predicted differences is approximately the same as that of predicted values. This is rather unexpected observation because while predicting the differences one eliminates the linear trend.

Above conclusions are based on comparatively small number of the real life data sets. In a future research, it would be desirable to make experiments using more sets of real life data and the artificial data sets, too. A special attention should be paid to cases when the prediction equations are non-linear with respect to input parameters.

References

Box, G.E.P., and G.M. Jenkins (1974). *Time Series Analysis. Forecasting and Control*. Mir, Moscow (in Russian).

Mockus, J. (1997). The set of examples of global and discrete optimisation, I. *Informatica*, **8**, 237–264.

Mockus, J. (1997). The set of examples of global and discrete optimisation, II. *Informatica*, **8**, 495–526.

Raudys, A. (1998). A non-parametric data mapping technique for active initialisation of the multilayer perseptron. In *Advances in Pattern Recognition. Springer Lecture Notes in Computer Science*, Vol. 1451. Springer Verlag. pp. 989–996 (Proc. Joint IAPR Int. Workshops SSPR'98 and SPR'98, Sydney, Australia, August 11–13, 1998).

Rumelhart, D.E., G.E. Hinton and R.J. Williams (1986). Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, I. Bradford Books, Cambridge, MA. pp. 318–362.

Soofi, A., and J. Mockus (1996). Long memory process and exchange rate forecasting. In *Proceedings of the Conference on Forecasting Financial Markets.* Chemical Bank and Imperial College, London.

A. Raudys graduated Vilnius University, Lithuania, in 1993 and was awarded bachelor degree. In 1999 he graduated Kaunas University of Technology, Lithuania and was awarded master degree. Currently he is working in Institute of Mathematics and Informatics in Data Analysis Departament.

J. Mockus graduated Kaunas University of Technology, Lithuania, in 1952. He got his Doctor habilitus degree in the Institute of Computers and Automation, Latvia, in 1967. He is a head of Optimal Decision Theory Department, Institute of Mathematics and Informatics, Vilnius, and professor of Kaunas University of Technology.

Daugiasluoksnio perceptrono ir ARMA modelio parametrų ekonominių laiko eilučių prognozavimo tikslumo palyginimas

Aistis RAUDYS, Jonas MOCKUS

Straipsnyje, naudojant šešių realių, daugiamačių, ekonominių laiko eilučių duomenis, lyginami daugiasluoksniu perseptronu ir ARMA modeliu paremti prognozavimo metodai. Parodoma, kad sunkiai prognozuojamuose uždaviniuose ARMA modelio tikslumas aukštesnis negu perseptrono, o gerai prognozuojamuose uždaviniuose – perseptrono tikslumas aukštesnis.