1 AN ADAPTIVE RETRAINING METHOD FOR THE EXCHANGE RATE FORECASTING

Emilian DOBRESCU[*]
Iulian NASTAC[**]
Elena PELINESCU[***]

Abstract

The paper advances an original artificial intelligence-based mechanism for specific economic predictions. The time series under discussion are non-stationary; therefore the distribution of the time series changes over time. The algorithm establishes how a viable structure of an artificial neural network (ANN) at a previous moment of time could be retrained in an efficient manner, in order to support modifications in a complex input-output function of financial forecasting. A "remembering process" for the former knowledge achieved in the previous learning phase is used to enhance the accuracy of the predictions.

The results show that the first training (which includes the searching phase for the optimal architecture) always takes a relatively long time, but then the system can be very easily retrained, as there are no changes in the structure. The advantage of the retraining procedure is that some relevant aspects are preserved (remembered) not only from the immediate previous training phase, but also from the previous but one phase, and so on. A kind of slow forgetting process also occurs; thus it is much easier for the ANN to remember specific aspects of the previous training instead of the first training.

The experiments reveal the high importance of the retraining phase as an upgrading/updating process and the effect of ignoring it, as well. There has been a decrease in the test error when successive retraining phases were performed.

JEL Classification: C45, C53, F47
Keywords: Neural Networks, Exchange Rate, Adaptive Retraining, Delay Vectors, Iterative Simulation

[*] *Professor, Senior Researcher, National Institute for Economic Research of Romanian Academy, edobrescu@rdslink.ro.*
[**] *Associate Professor, Polytechnic University of Bucharest, nastac@ieee.org.*
[***] *Senior Researcher, Institute of Economic Forecasting, Romanian Academy, Elena.Pelinescu@itcnet.ro.*

1. Introduction

After 1989 Romania, as other Central and East European Countries, has experimented different exchange rate regimes, from the dual exchange rate at the beginning of the transition period to the managed floating exchange rate. Due these circumstances and the changing economic environment specific to the restructuring processes, the evolution of the exchange rate was characterised by a high volatility. Under such conditions, the artificial intelligence techniques could be a useful instrument in macroeconomic analysis and prediction.

Artificial neural networks (ANNs) have been widely applied to forecasting problems (Nastac 2004, Huang and Lewis 2003, Zhang *et al*. 1998). There is considerable interest in the development of reliable forecasting models for financial applications. Models based on the ANN have been found to be suitable for certain applications where other techniques failed. The idea that the ANN can be used for a better understanding of the economic complex mechanisms is present in the literature (Salzano 1999, Shadbolt 2002, Zhang 2003).

The goal of our research was to find a practical mathematical model that describes the relationship between 9 input variables and one output variable that model the USD exchange rate. All inputs and the output vary dynamically, and different time-delays might occur. Changing an input variable may result in an output change that starts only a day later and goes on for up to several days.

The gross data consist of 2310 rows (time steps) – one data row every working day during 9 years (January 3, 1995 – January 30, 2004). The time series under discussion are nonstationary. The nonstationary characteristic implies that the distribution of the time series changes over time. The recent data points could provide more important information than the distant data points. Therefore, we propose a new adaptive retraining mechanism to take this characteristic into account.

This paper is organized as follows. Section 2 presents the issue that concerns the model structure and data preprocessing. In the next section, we introduce the adaptive retraining technique and explain our approach. The main features of the experimental results are given in Section 4. Finally, Section 5 concludes the paper.

2. The Model Architecture

The time-delay or deadtime is frequently encountered in the financial systems. It is well known that feedback control in the presence of time-delay leads to particular difficulties, since a delay places a limit on the time interval.

In Fig. 1 we present our idea of training a feedforward ANN such that the latter becomes a predictor. We use delayed rows of the nine input data (see the next section) to simulate the current states of the USD exchange rate. For learning purposes, the network inputs involve many blocks with several time-delayed values of the financial system inputs, and fewer blocks with system delayed output. The ANN target-output consists of the current value of the corresponding USD exchange rate.

Therefore, the system tries to match the current values of the output, by properly adjusting a function of the past values of the inputs and output (Fig. 1).

At the current moment, t, the output (Fig.1) is affected by the inputs at different previous timesteps ($t - i\_d_1$, …, $t - i\_d_n$), and also by the outputs at other previous timesteps ($t - o\_d_1$, …, $t - o\_d_m$), respectively. We denote by the so-called *In_Del* and *Out_Del*, two *delay vectors* that include the delays that we take into account:

$$In\_Del = \left[i\_d_1, i\_d_2, ..., i\_d_n\right] \quad (1)$$

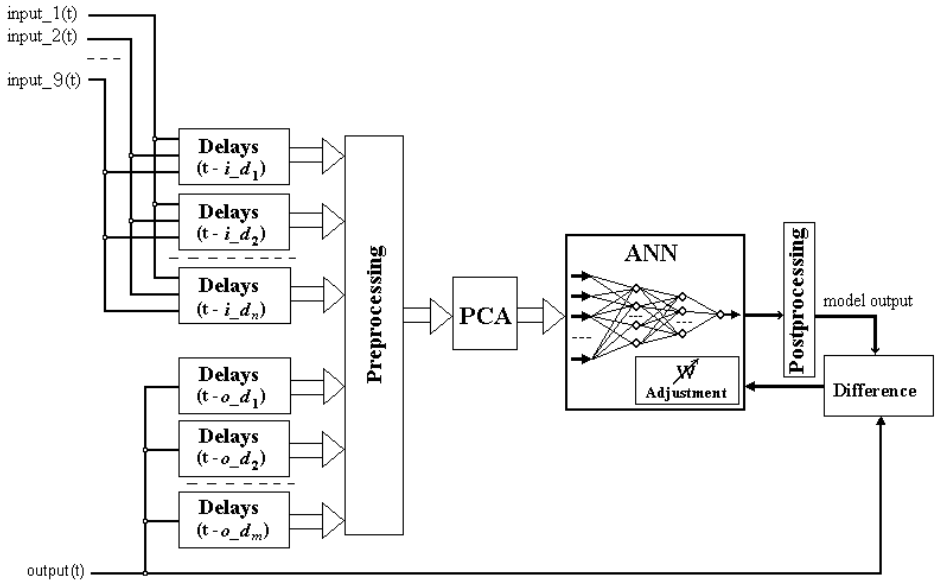
and

$$Out\_Del = \left[o\_d_1, o\_d_2, ..., o\_d_m\right] \quad (2)$$

where n > m.

Figure 1

The forecasting architecture. The training process



For *In_Del*, we use various delay vectors with *n* = 7, 8 or 9 elements, whose values are within a range of twenty days. Regarding *Out_Del*, we employ different combinations, with *m* = 3, 4 or 5 elements, covering about one week. The distribution of the vector elements is preferably (but not compulsory) chosen similar to the Gamma distribution. The elements of each vector are ascendingly ordered. Consequently, the maximum values of any delay vector are i_d_n or o_d_m, respectively. The recurrent relation performed by the model is as follows:

(3)

where X is the input vector; $i = 1,...,n$ and $j = 1,...,m$.

We use feedforward ANNs with two hidden layers in order to achieve a good approximation function, based on our preliminary research, where we have obtained better results in the case of two hidden layers than in the case of one hidden layer, however maintaining a similar ratio (approx. 5/1) between the number of the training samples and the total number of the weights. The ANN models, depicted in Figure 1, use training sets of $V\text{-}i\_d_n$ input-output pairs for model adaptation (see the next section), where V = 2240 is the initial time steps interval employed for the training purpose.

Once we have established all the influences on the output at the moment *t*, we apply Principal Component Analysis (PCA) (Jackson 1991) to reduce the dimensionality of the input space and to un-correlate the inputs. Before applying PCA, we preprocessed the inputs and outputs, by replacing the missing data using the previously available values and, then, by applying the normalization. Data preprocessing prepares the raw data for the forecasting model and transforms it into a format that will be easier and more effectively processed. Finally, we have applied the reverse process of normalization, in order to denormalize the simulated outputs. Data preprocessing and data post-processing are essential steps of the knowledge discovery process in the real world applications, and they greatly improve the network's ability to capture valuable information, if they are correctly carried out (Hagan et al. 1996, Basheer *et al*., 2000).

Our attempt will involve the following variables:

- Daily exchange rate ROL/US Dollar (CSZ_USD);
- Monthly foreign assets, in months of imports, transformed on the daily basis by keeping constant every day the month value (ACIMP);
- Monthly changes in the consumer price index (ÄCPI), transformed on the daily basis by keeping constant every day the month value ;
- Daily value of the purchases (of foreign currency) from the clients (PVZ_CC);
- Daily value of the purchases (of foreign currency) from the commercial banks (PVZ_CA);
- Daily value of the purchases (of foreign currency) from the central bank (PVZ_CB);
- Daily value of the sales (of foreign currency) to the clients (PVZ_VC);
- Daily value of the sales (of foreign currency) to the commercial banks (PVZ_VA);
- Daily value of the sales (of foreign currency) to the central bank (PVZ_VB);
- Month indicator L (the days of January are denoted by 1, the days of February by 2 and so on).

The period covered by analysis is from January 3, 1995 (the first year after a major liberalisation of the forex market) to January 30, 2004.

The possible connection between the exchange rate and other mentioned variables has been checked using the Granger causality tests (Granger 1969, Granger 1988), which were computed for different number of lags (starting from 26). As it was expectable for a daily analysis, such an interdependence is clearly revealed in the case of 1-2 lags (Tables 1 and 2).

Table 1

The Granger Causality Test (Two Lags)

Sample: 1/03/1995 1/30/2004

Null Hypothesis for two lags:	Obs	F-Statistic	Probability
CSZ_USD does not Granger Cause ACIMP	2308	6.93688	0.00099
ACIMP does not Granger Cause CSZ_USD		4.47268	0.01152
CSZ_USD does not Granger Cause CPI	2308	2.17540	0.11380
CPI does not Granger Cause CSZ_USD		1.73526	0.17659
PVZ_CA does not Granger Cause CSZ_USD	2308	1.37646	0.25268
CSZ_USD does not Granger Cause PVZ_CA		179.612	0.00000
PVZ_CB does not Granger Cause CSZ_USD	2308	0.69241	0.50047
CSZ_USD does not Granger Cause PVZ_CB		16.3405	9.0E-08
PVZ_CC does not Granger Cause CSZ_USD	2308	10.0511	4.5E-05
CSZ_USD does not Granger Cause PVZ_CC		99.5828	0.00000
PVZ_VA does not Granger Cause CSZ_USD	2308	0.16233	0.85017
CSZ_USD does not Granger Cause PVZ_VA		156.138	0.00000
PVZ_VB does not Granger Cause CSZ_USD	2308	2.91803	0.05424
CSZ_USD does not Granger Cause PVZ_VB		26.3544	4.8E-12
PVZ_VC does not Granger Cause CSZ_USD	2308	5.67277	0.00349
CSZ_USD does not Granger Cause PVZ_VC		67.5337	0.00000

Table 2

The Granger Causality Test (One Lag)

Null Hypothesis for one lags:	Obs	F-Statistic	Probability
CSZ_USD does not Granger Cause ACIMP	2309	13.6943	0.00022
ACIMP does not Granger Cause CSZ_USD		10.2776	0.00136
CSZ_USD does not Granger Cause CPI	2309	2.57251	0.10887
CPI does not Granger Cause CSZ_USD		4.02186	0.04503
PVZ_CA does not Granger Cause CSZ_USD	2309	0.15974	0.68944
CSZ_USD does not Granger Cause PVZ_CA		705.430	0.00000
PVZ_CB does not Granger Cause CSZ_USD	2309	0.01291	0.90955
CSZ_USD does not Granger Cause PVZ_CB		0.79380	0.37305
PVZ_CC does not Granger Cause CSZ_USD	2309	20.8006	5.4E-06
CSZ_USD does not Granger Cause PVZ_CC		424.617	0.00000
PVZ_VA does not Granger Cause CSZ_USD	2309	0.13588	0.71245
CSZ_USD does not Granger Cause PVZ_VA		639.103	0.00000
PVZ_VB does not Granger Cause CSZ_USD	2309	0.00294	0.95676
CSZ_USD does not Granger Cause PVZ_VB		54.9414	1.7E-13
PVZ_VC does not Granger Cause CSZ_USD	2309	1.61792	0.20351
CSZ_USD does not Granger Cause PVZ_VC		288.685	0.00000

Tables 1 and 2 show that an artificial neural network based on the previously mentioned variables can be considered as economically consistent.

3. The Adaptive Retraining Procedure

The feature of universal functional approximator (Hornik *et al*., 1989) adds the power and flexibility of the neural networks to the process of learning complex patterns and relationships. However, the potential risk of using the universal approximator is the overfitting problem, since it is often easy to train a large network model to learn the peculiarities, as well as the underlying relationship. Therefore, the balance between the learning capability and the generalization power is very important in the neural network forecasting applications.

As basic training algorithm, we use the Scale Conjugate Gradient (SCG) algorithm (Moller 1993). In order to avoid the overfitting phenomenon, we apply the early stopping method (*validation stop*) (Hagan *et al*., 1996) during the training process.

Next, the adaptivity of the result is performed (and improved), by using the *retraining technique* (Nastac, 2004, Nastac and Matei, 2003), in a special way. This technique is a mechanism for extracting practical information directly from the weights of a reference ANN that was already trained in a preliminary phase. The retraining procedure reduces the reference network weights (and biases) by a *scaling factor* γ, $0<\gamma<1$. The reduced weights are further used as the initial weights of a new training sequence, with the expectation of a better accuracy.

Briefly, the entire technique can be summarized by the following phases:

- Training an Artificial Neural Network in a natural way with *validation stop,* and with the weights initialized to small, uniformly distributed values;
- Reducing the first network weights and biases by a scaling factor γ $(0 < \gamma < 1)$;
- Retraining the network with the new initial weights;
- Comparing the *validation error* (or training error) in both cases.

An advantage of this technique is a significant decrease in the number of training cycles, as compared to the classical training methods (Nastac and Matei, 2003).

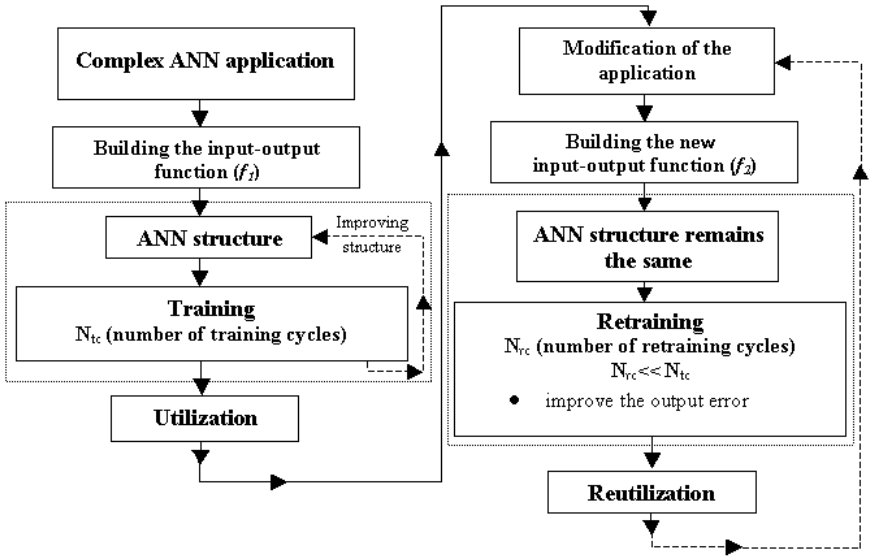
The retraining technique allows us to improve continuously the model, at times, by using new (shifted) databases (see Figure 2).

The data that we have used in our model consist of $V\text{-}i\_d_n$ input-output pairs during each training (or retraining) phase, where V = 2240 (January 3, 1995 – October 17, 2003) is the initial time steps interval employed for the training purpose. As a splitting criterion, we randomly choose approximately 85% of the data ($V\text{-}i\_d_n$) for the training set, and the rest for validation. Furthermore, we imposed the supplementary condition:

(4)

Figure 2

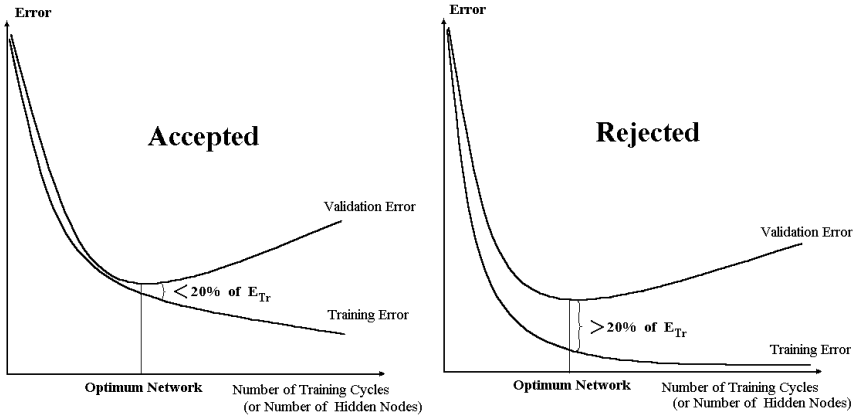
The retraining technique (step by step)



to avoid a large difference (see Figure 3) between the error of the training set (E_{tr}) and the error of the validation set (E_{val}). In this way, the overfitting phenomenon on the test set will be considerably reduced. In our approach, the validation set acts at the same time as a kind of test set, although there is a real and separate test set of T = 20 different and successive timesteps (where T<<V).

Figure 3

A supplementary condition over the validation set



Next, we describe the *steps* that we have taken to adapt our model:

1. Firstly, we set the proper number of hidden neurons for each hidden layer (N_{h1} and N_{h2}). Each of the training sessions started with the weights

initialized to small uniformly distributed values (Hagan *et al*., 1996, Nastac and Matei, 2003). We tested several pyramidal ANN architectures, with N_{h1} and N_{h2} taking values in the vicinity of the geometric mean (Basheer *et al*., 2000) of the neighboring layers, and observing the following rules:

(5)

(6)

(7)

Above, N_i is the number of inputs after the PCA block and N_o is the number of the outputs (N_o=1). Each architecture was tested five times, with random initial settings of the weights and different training/validation sets. We chose the best model with respect to the smallest error between the desired and simulated outputs. This error (E_{tot}) was calculated for V-i_d_n data that included both training and validation sets.

2. Secondly, we predicted the T values of the outputs (during the interval (V+1) - (V+T)), in a sequential mode. Let us call this step the **Iterative Simulation** (IS) of the output. Therefore, in order to produce one output at timestep *t*, the neural network used as input the estimated outputs (besides the real inputs) that were calculated at the previous steps, by using other simulated outputs, and so on. Applying this iterative process, a forecast may be extended as many steps as required, nevertheless taking the risk that each step increases the forecasting error.

   Then, we computed the error *ERR* (Nastac 2004) that represents the accuracy of the approximation of the output data, within the forecasting horizon of *T* timesteps:

(8)

   where *T* = number of timesteps (working days)

   O_{Rp} = real output at timestep *p*

   O_{Fp} = forecasted output at timestep *p*

   and a weight function that decreases with the number of timesteps *p*.

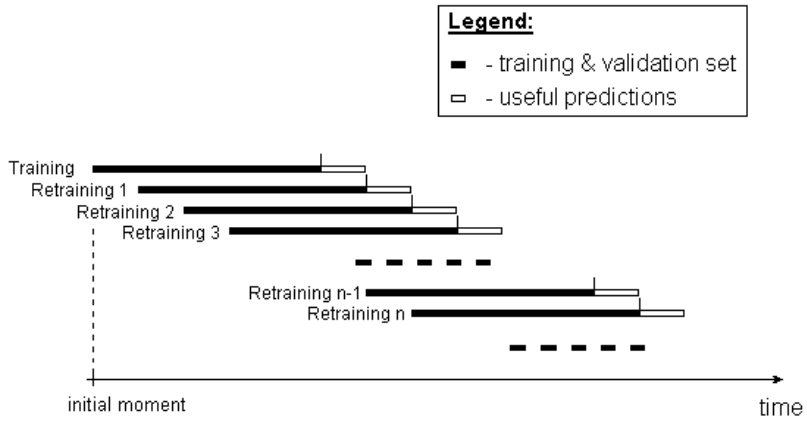
3. Thirdly, we applied the retraining technique for a shifted interval of timesteps (*Shift*+1) – (*Shift*+*V*), where *Shift* $\leq$ *T*. Here we used the ANN architecture that resulted at the end of the previous step. We applied this technique for each value of γ (γ = 0.1, 0.2, …, 0.9), keeping the neural network (weight distribution) that achieved the minimum error, as the reference network. We repeated this step five times, and we randomly reconstructed the training and validation sets each time.

4. Fourthly, we predicted the *T* values of the outputs (during the interval of timesteps (*Shift*+*V*+1) - (*Shift*+*V*+*T*)), in the same sequential mode as in step 2 (Iterative Simulation).
5. We repeated *L* times the steps 3 and 4 at successively shifted intervals of *V* timesteps for retraining processes and *T* timesteps for sequential forecasting. Each time the intervals were ascendingly repositioned with *Shift* timesteps (days).

Firstly, a decisive role in choosing the best model is played by the mean square error of the differences between the real and the simulated outputs of V-i_d_n data rows, which included both the training and the validation sets. Afterwards, the retraining technique adapts the ANN system, in order to learn continuously the latest evolution of the financial process. The retraining process can be viewed as a "remembering process" of the former knowledge achieved in the previous learning phases. Figure 4 illustrates the evolution of the trainings.

Figure 4

The training and retraining phases



The retraining technique allows us to improve continuously the model, at times, by using new (shifted) databases (see Figure 4). For a single combination of the delay vectors, we obtained (and used) a model with its associated adaptive behavior. The above-mentioned steps were applied to different delay vectors.

4. Experimental Results

We performed the steps described in Section 3 for various combinations of delay vectors. Each time, the first step, which determines the optimum architecture, required a somewhat longer time (approx. one day). The optimum architecture highly depends on the delay vectors. Then, for each retraining phase the program worked for about one hour. There was a clear difference between the first training process, which

needed a long time to search for the best architecture, and the retraining on the other hand. It can be quite easy to retrain a good ANN architecture several times, by using a shifted training set. In Table 3 we present the values of the test error (*ERR* according to (8)) for iterative simulations of the output, computed at the end of the first training and, then, after each successive retraining phase (L = 20), when using:

Case I: *In_Del* = [1 2 3 4 5 6 8 12] and *Out_Del* = [0 1 2 4].

Case II: *In_Del* = [2 3 4 5 6 7 9 13] and *Out_Del* = [0 1 2 4].

Case III: *In_Del* = [3 4 5 6 7 8 10 14] and *Out_Del* = [0 1 2 4].

Case IV: *In_Del* = [4 5 6 7 8 9 11 15] and *Out_Del* = [0 1 2 4].

Case V: *In_Del* = [5 6 7 8 9 10 12 16] and *Out_Del* = [0 1 2 4].

We carried out the simulations under the following assumptions:

- V = 2240 working days are enough for the first training phase and then for each retraining phase;
- T = 20 working days represents the prediction horizon;
- *Shift* = 1 working day is the shifting time for the next retraining.

It is worth mentioning that the values of the previous parameters can be easily changed. The prediction horizon (T) can be, for example, enlarged to 50 timesteps (see Figure 9 and Figure 10) or more. Choosing the number of samples for training is an open issue: not too small to have enough data (more than five times the number of samples versus the number of weights), but not too large, especially in a nonstationary environment.

The last three lines of Table 3 represent the results of using the systems obtained after retraining 20 under three special situations:

- Computing ERR when PVZ_CB is missing (always zero) from the test set of 20 working days;
- Computing ERR when test set is enlarged to 50 working days;
- Computing ERR when PVZ_CB is missing (always zero) from the test set of 50 working days.

Obviously, ERR increase when the test set is enlarged to 50 timesteps. The effect of missing the PVZ_CB consist of an expected increase in ERR (except Case I !).

Table 3

Evolution of Test Error (Iterative Simulations of Output) during training and retraining phases

	Training / retraining interval	Test interval	Test Error (Case I)	Test Error (Case II)	Test Error (Case III)	Test Error (Case IV)	Test Error (Case V)

	Training / retraining interval	Test interval	Test Error (Case I)	Test Error (Case II)	Test Error (Case III)	Test Error (Case IV)	Test Error (Case V)
First training	1 – 2240	2241 – 2260	1.1888	1.2949	1.2108	1.2898	1.3928
Retraining 1	2 – 2241	2242 – 2261	1.1892	1.3371	1.2709	1.2017	1.2158
Retraining 2	3 – 2242	2243 – 2262	1.3926	1.1909	1.299	1.3673	1.2948
Retraining 3	4 – 2243	2244 – 2263	1.2237	12.12	11.704	1.4511	1.3793
Retraining 4	5 – 2244	2245 – 2264	1.4383	1.3095	1.5615	1.5651	1.5873
Retraining 5	6 – 2245	2246 – 2265	1.5788	1.4243	1.5182	1.6042	1.3821
Retraining 6	7 – 2246	2247 – 2266	1.76	1.6569	1.7775	1.8596	1.6161
Retraining 7	8 – 2247	2248 – 2267	1.7488	1.8773	1.8477	1.9052	1.9589
Retraining 8	9 – 2248	2249 – 2268	1.9299	1.9781	1.9314	1.4638	1.9754
Retraining 9	10 – 2249	2250 – 2269	1.9097	1.6711	1.2467	1.7996	1.8251
Retraining 10	11 – 2250	2251 – 2270	1.7659	1.5293	1.346	1.4897	1.3922
Retraining 11	12 – 2251	2252 – 2271	1.6921	1.3387	1.2109	1.2601	1.2665
Retraining 12	13 – 2252	2253 – 2272	1.15	0.87714	0.87868	1.0243	0.98031
Retraining 13	14 – 2253	2254 – 2273	0.554	0.66446	0.67551	0.73437	0.73749
Retraining 14	15 – 2254	2255 – 2274	0.56147	0.60283	0.53602	0.55817	0.55777
Retraining 15	16 – 2255	2256 – 2275	0.51149	0.45048	0.53535	0.56274	0.53825
Retraining 16	17 – 2256	2257 – 2276	0.7822	0.43994	0.54174	0.498	0.62045
Retraining 17	18 – 2257	2258 – 2277	1.0197	0.67071	0.58516	0.64491	0.70827
Retraining 18	19 – 2258	2259 – 2278	0.49848	0.74687	0.83123	0.58157	0.71187
Retraining 19	20 – 2259	2260 – 2279	0.42343	0.69114	0.82119	0.60837	0.72988

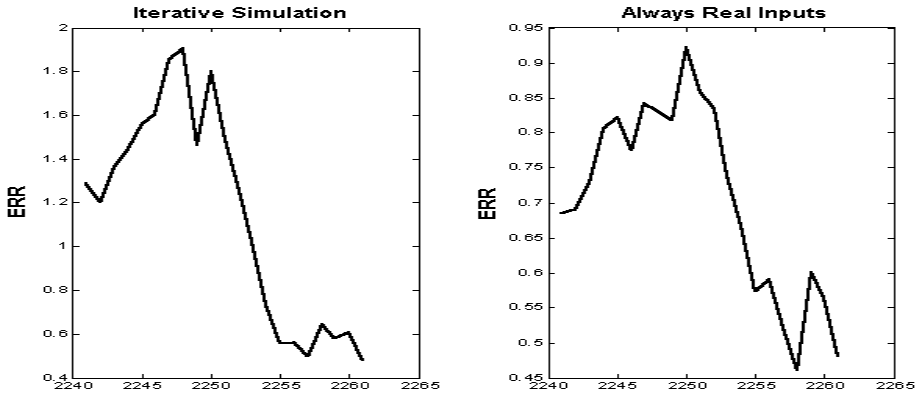
	Training / retraining interval	Test interval	Test Error (Case I)	Test Error (Case II)	Test Error (Case III)	Test Error (Case IV)	Test Error (Case V)
Retraining 20	21 – 2260	2261 – 2280	0.46369	0.38158	0.53764	0.48056	0.55116
Retraining 20 (prediction without PVZ_CB	21 – 2260	2261 – 2280	0.4384	0.4575	0.5756	0.4764	0.6000
Retraining 20 (long prediction: T=50)	21 – 2260	2261 – 2310	1.0059	0.6652	1.4908	1.0295	1.3860
Retraining 20 (long prediction without PVZ_CB)	21 – 2260	2261 – 2310	0.9116	0.8916	1.9488	1.1198	1.5520

A natural question is: "What is the matter without retraining?". We made an experiment and we omitted to retrain the system during the retraining phase 3 (the training interval 4 – 2243 see Table 3) in the Case II and Case III. We can easily see in Table 3 that ERR had a high value for the corresponding test interval, when the retraining was not performed, but then the ERR came back to the normal range at the next intervals, when successive retraining procedures were performed. This experiment reveals the high importance of the retraining phase as an upgrading/updating process and the effect of ignoring it, as well.

In Table 3, it seems that the delay vectors have been properly chosen, since, finally, there has been a decrease in the test error when successive retraining phases were performed. An example is the Case IV (see Figure 5) of ERR, for L = 20 successive retraining phases (note that the abscissa represents the successive first values of the related test set). Figure 5 includes two graphs of the ERR: the left one shows the evolution of the ERR when **Iterative Simulation** (IS) is employed (see the second step from the previous section), and the right one when the system "**Always uses the Real Inputs"** (ARI), which included the real previous outputs and not the estimated ones. It is obviously that the Iterative Simulation provides higher values of the ERR as compared to the situation when the system is always fed with real inputs, but often the difference is not significant. Practically, a long term forecasting can be implemented using only the Iterative Simulation and the second approach remains an utopia.

Figure 5

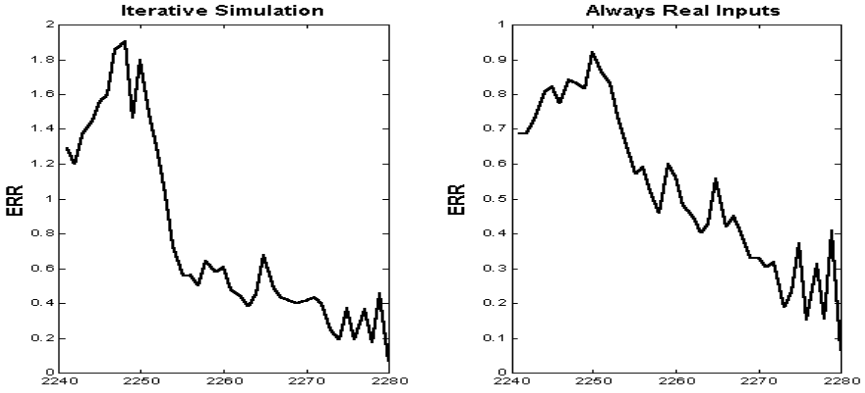
ERR trend (Case IV) of test sets for the first training and *L* = 20 successive retraining phases



An extension of the same experiment (Case IV) is shown in Figure 6, for *L* = 40 successive retraining phases, in order to prove the global decreasing trend of the ERR.

Figure 6

ERR trend (Case IV) of test sets for the first training and *L* = 40 successive retraining phases



The quality of the predictions can be also graphically analyzed, by enforcing a tube around the real outputs, given by a function like the one below:

(9)

Here, *A* is an acceptable prediction error, *q* is an increasing factor and *n* is the number of predicted timesteps. The predicted output values should lay in the interval output(*n*) +/- *f*(*n*), represented by the dotted lines in the figures 7-10.

Figure 7

Data forecasting for the test interval of retraining 20 (Case II)
ERR_{IS} = 0.38158 and ERR_{ARI} = 0.37344

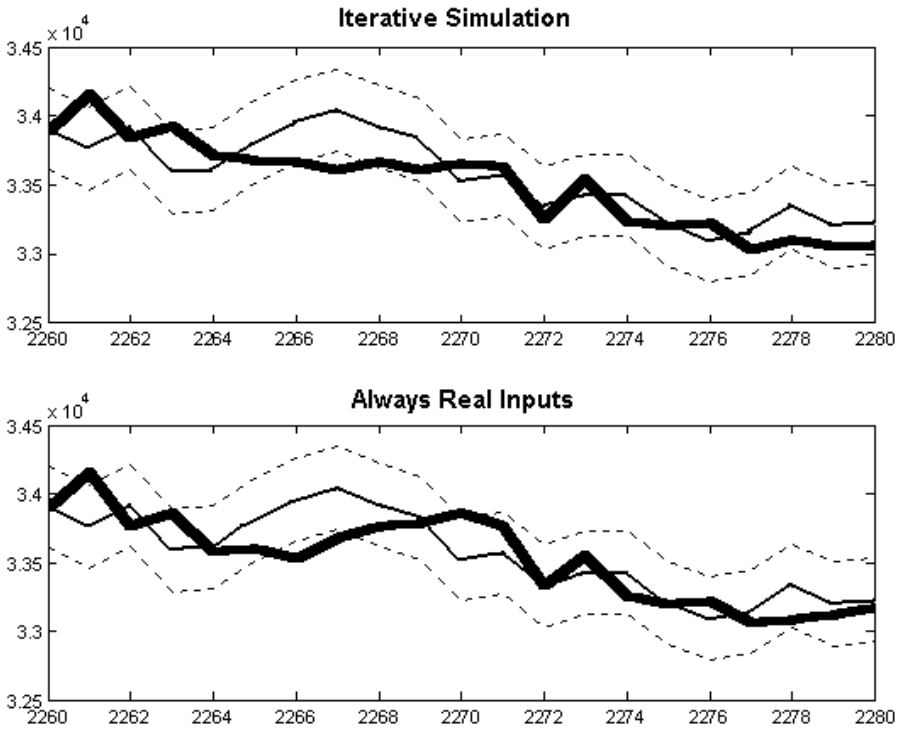


Figure 7 shows the graphs of the USD exchange rate (Case II) for the last test interval (of retraining 20). The real data are represented with thin lines and the neural network output values with thick lines. There are two graphs in the same figure: the first shows the evolution of the output when Iterative Simulation (IS) is employed and the second one when the system *Always* uses *Real Inputs* (ARI). There is a "tube" (dotted lines) around the real data, given by the function $f(n)=300+0.05 \cdot n$ (where n = 1 … 20).

Note that the graphs are extended to the left with one more value that it corresponds to the last value of the training/validation interval. This way, both lines (thin and thick) start from the approximate same point. The difference between ERR_{IS}=0.38158 and ERR_{ARI}=0.37344 is not significant.

The next three figures show the similar results (of Case II) when the systems obtained after retraining 20 is used under three special circumstances (already mentioned):

- Graphs of the simulations when PVZ_CB is missing from the test set of 20 working days (Figure 8);
- Graphs of the simulations when the test set is enlarged to 50 working days (Figure 9);
- Graphs of the simulations when PVZ_CB is missing from the test set of 50 working days (Figure 10).

Figure 8

Data forecasting for the test interval of retraining 20 (Case II) when PVZ_CB is missing from the test set

$ERR_{IS} = 0.4575$ and $ERR_{ARI} = 0.4159$

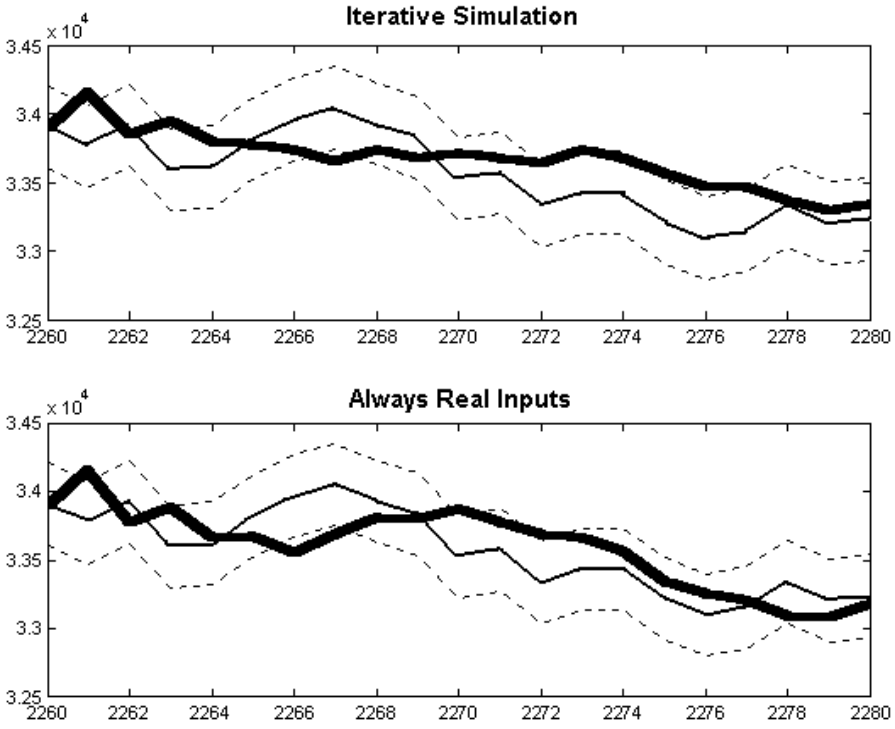
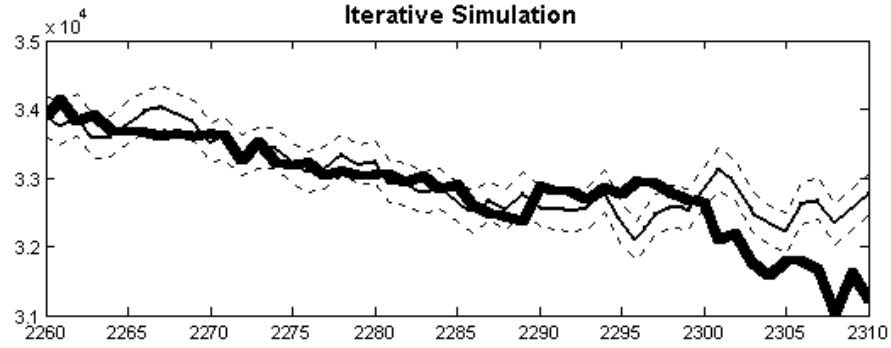


Figure 9

Data forecasting for the test interval of retraining 20 (Case II) when the test set is enlarged to 50 working days.

ERR_{IS} = 0.6652 and ERR_{ARI} = 0.5077



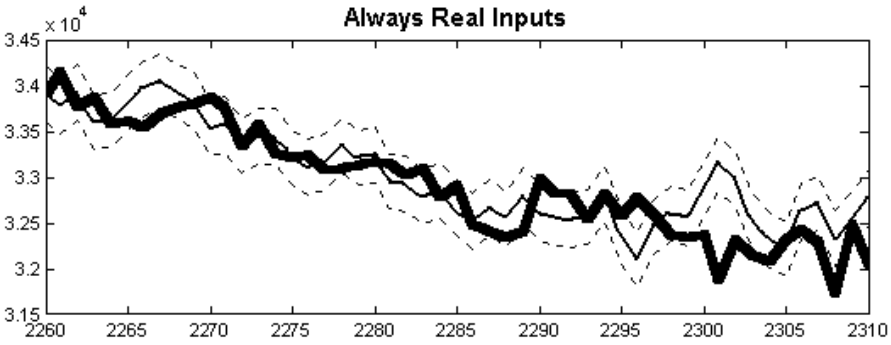
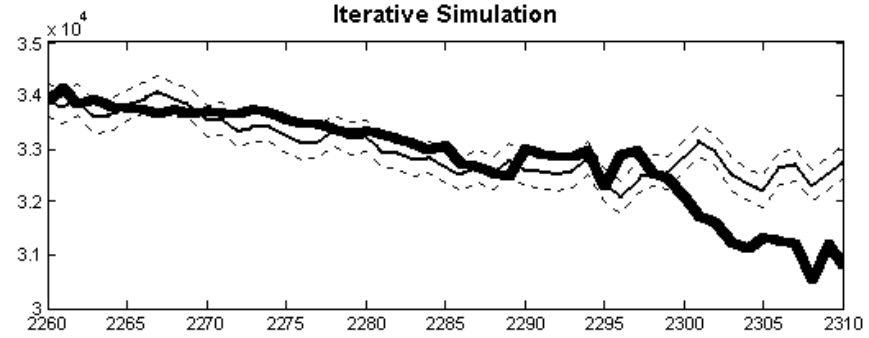
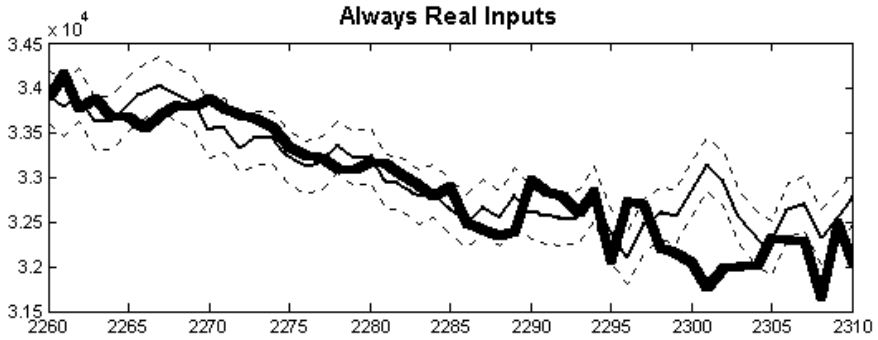


Figure 10

Data forecasting for the test interval of retraining 20 (Case II) when PVZ_CB is missing from the test set of 50 working days.

ERR_{IS} = 0.8916 and ERR_{ARI} = 0.5780





We noticed that, except for a few cases, in almost all the graph representations of the predictions the trends were well captured (even outside of the "tube") by using our approach. Moreover, we showed both graphs (with Iterative Simulation and when the system always uses the real inputs) in order to demonstrate the soundness of our approach. The Iterative Simulation does not increase the error as much as one could expect at the first sight. Nevertheless, the difference between ERR_{IS} and ERR_{ARI} becomes clearer as the length of test set increases. The long-term prediction is not very accurate as long as after a while the simulated outputs evidently exceed the limits of the "tube" around the real outputs (Figure 9 and Figure 10).

5. Conclusions

The ANNs ability to extract significant information from its training data provides a valuable framework for the representation of relationships that are present in the structure of the data. This allows for both the interpolation among the *a priori* defined points and the extrapolation outside the range bordered by the extreme points of the training set.

The evaluation of the test error shows that the adaptive retraining technique can gradually improve, on the average, the achieved results. Our practical experience reveals that the first training (which includes the searching phase for the optimal architecture) always takes a relatively long time, but then the system can be very easily retrained, as there are no changes in the structure. The great advantage of the retraining technique is that some relevant aspects are preserved (*remembered*) not only from the immediate previous training phase, but also from the last but one phase, and so on. A kind of *slow forgetting process* also occurs; thus it is much easier for the ANN to remember specific aspects of the previous training instead of the first training. It means that the former information accumulated during the previous trainings will be slowly forgotten and the learning process will be adapted to the newest evolutions of the financial process.

The final remark refers to the basic training algorithm. Even if the SCG algorithm is not the fastest algorithm, the great advantage is that this technique works very efficiently

for networks with a large number of weights. The SCG is something of a compromise; it does not require large computational memory, and yet it still has a good convergence and is very robust. Furthermore, we always apply the early stopping method (validation stop) during the training process, in order to avoid the over-fitting phenomenon. In addition, it is well known that for the early stopping one must be careful not to use an algorithm that converges too rapidly. The SCG is properly suited for the validation stop method. Nevertheless, it is quite easy to replace the SCG algorithm with another one, since the adaptive retraining technique is flexible and independent of the basic training algorithm.

References

Basheer, I.A., and Hajmeer, M., (2000) "Artificial neural networks: fundamentals, computing, design, and application", *Journal of Microbiological Methods*, Elsevier Science, Vol. 43, pp. 3-31.

Bordo M. D., (2003), *Exchange Rate Regime Choice in Historical Perspective*, IMF, Working Paper, 03/160, International Monetary Fund, Washington DC.

Dickey, D. and S. Pantula, (1987), "Determining the Order of Differencing in Autoregressive Process", *Journal of Business and Economic Statistics* 5, pp. 455 - 461.

Dickey, D., Hasza D.P., and Fuller W.A. (1984), "Testing for unit roots in seasonal time series", *Journal of the American Statistical Association* 79, pp. 355-367.

Dobrescu, E., (1998), *Macromodels of the Romanian Transition Economy*, The Expert Publishing House, Bucharest.

Granger, C.W.J., (1969), "Investigating Causal Relations by Econometric Models", *Econometrica*, 37, pp. 424-438.

Granger C.W.J., (1988), "Some Recent Developments in a Concept of Causality", *Journal of Econometrics*, 39, pp. 199-211.

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

Hornik, K., Stinchcombe, M., and White, H., (1989), "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol.2, pp. 359-366.

Huang, J. Q., and Lewis, F. L., (2003) "Neural-network predictive control for nonlinear dynamic systems with time-delay", *IEEE Trans. Neural Networks*, vol. 14, pp. 377-389.

Jackson, J.E., (1991), *A user guide to principal components*, John Wiley, New York.

Moller, M.F. "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, pp. 525-533, 1993.

Mussa M. (1986), "Nominal Exchange Rate Regimes and the Behavior of Real Exchange Rates, Evidence and Implications", *Carnegie-Rochester Conference Series on Public Policy*, Vol. 25, pp. 117-213.

Nastac, I., (2004) "An Adaptive Retraining Technique to Predict the Critical Process Variables", *TUCS Technical Report*, No. 616, June 2004, Turku, Finland.

Nastac, I., and Matei, R., (2003) "Fast retraining of artificial neural networks," in *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, Wang *et al*. (Eds.), Springer-Verlag in the series of *Lecture Notes in Artificial Intelligence* (LNAI 2639), pp. 458-462.

Pelinescu, E. (2003) "FEER: Relevant theoretical and empirical literature", *Romanian Journal of Economic Forecasting*, Spec. Edition 1., The Expert Publishing House, Bucharest, pp. 111-118.

Salzano, M., (1999) "Neural Networks as tools for increasing the forecast and control of complex economic systems", *Economics & Complexity*, Vol. 2, No. 2.

Shadbolt, J. (Ed.), and Taylor, J.G. (Ed.) (2002), *Neural Networks and the Financial Markets*, Springer-Verlag Telos.

Zhang, G., Patuwo, B.E., and Hu, M.Y., (1998), "Forecasting with artificial neural networks: The state of the art," *Int. J. Forecasting*, vol. 14, pp. 35-62.

Zhang, G.P., (2003) "Neural Networks in Business Forecasting", *Information Science Publishing*.