A flexible forecasting intelligent model for nonstationary time series

Iulian Nastac

Faculty of Electronics, Telecommunications and Information Technology
POLITEHNICA University of Bucharest
OP 16, CP 77, 061112 Bucharest, ROMANIA
nastac@ieee.org

Abstract

The paper presents a general adaptive model for those dynamic systems that work on continuously changing environments. This interdisciplinary model is tested in the financial, genetic and technical fields. The algorithm of the model establishes how a viable structure of an artificial neural network 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 a real forecasting system. A “remembering process” from the previous learning phase is used to enhance the accuracy of the predictions. The advantage of the retraining procedure is that some relevant aspects are preserved 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 model to remember specific aspects of the previous training instead of an oldest one.

Keywords: neural networks, forecasting, time series, retraining

1. Introduction

In the scientific community there is a considerable interest on the development of new reliable forecasting models. The classical approaches (Armstrong, 2001) of data prediction/forecasting (regression analysis, econometrics models, Pareto analysis, etc.) proved many times their limitations, being surpassed by new methods and techniques based on artificial intelligence models (Heravi et al., 2004) (Chen et al., 2008) (Lin and Lee, 2009). Although the Artificial Intelligence Theory has already a large number of applications, the forecasting of time series is relatively recent a presence increasingly sustained in specific publications. Interest in neural network forecasting has resulted in a tremendous number of applications reported during the last decade (Berardi and Zhang, 2003) (Wang et al., 2009).

Artificial neural networks (ANNs) have been widely applied to forecasting problems (Huang and Lewis, 2003) (Nastac et al. 2007) (Sousa et al., 2007) (Pino et al., 2008) (Weckman et al. 2008) since they are less susceptible to the problem of misspecification as compared to most parametric models. ANNs are also more noise tolerant, having the ability to learn complex systems with incomplete or corrupted data. In this paper, the time series and sequences under discussion are inherently noisy and nonstationary. The nonstationary characteristic implies that the distribution of the time series changes over time. Furthermore, some gradual changes in the dependency between the input and
output variables may appear. In other words, the recent data points could provide more important information than the distant data points. Therefore, we propose an adaptive retraining mechanism to take this characteristic into account and prove its worth on various applications.

This paper is organized as follows. Section 2 presents the problem 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 sections 4 and 5. Finally, Section 6 concludes the paper.

2. General structure approach

Usually, time-delays are frequently encountered in financial or technical systems. A marked spatial correlation is also presented in some applications, like DNA genomic sequences. It is well known that feedback control in the presence of time/spatial delay leads to particular difficulties, since a delay places a limit on the time/spatial interval.

In order to cover a wide number of possible applications, we start to consider a system with $s$ inputs and $u$ outputs. 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 input data to simulate the current states. For learning purposes, the network inputs involve many blocks with several time-delayed values of the system inputs, and fewer blocks with system delayed outputs. Each ANN target-output consists of the current value of the corresponding sequence. 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 outputs.

![Fig. 1. Forecasting architecture. Training process.](image)

At the current moment $t$, the output output $k$ ($k = 1 \ldots u$, Fig. 1) is affected by the inputs at different previous time/space steps ($t - i \cdot d_i$, $\ldots$, $t - i \cdot d_a$), and also by the outputs at other previous time/space steps ($t - o \cdot d_i$, $\ldots$, $t - o \cdot d_m$), respectively. We denote by so called In Del and Out Del, two delay vectors that include the delays that we take into account:
\[ \ln \_ \text{Del} = [i \_ d_1, i \_ d_2, \ldots, i \_ d_n] \]  
(1)

\[ \text{Out} \_ \text{Del} = [o \_ d_1, o \_ d_2, \ldots, o \_ d_m] \]  
(2)

where \( n > m \).

The distribution of the vector elements is usually (but not compulsory) 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. Usually, \( i \_ d_n \) significantly exceeds \( o \_ d_m \).

The recurrent relation performed by the model that predicts the output \( k \) is as follows:

\[ y_k(t+1) = \tilde{F}(X(t+1 - \ln \_ \text{Del} i), Y(t - \text{Out} \_ \text{Del} j)) \]  
(3)

where \( X \) is the input vector, \( Y \) the corresponding output vector, \( y_k \) the value of output \( k \), \( i = 1, \ldots, n \) and \( j = 1, \ldots, m \).

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 case of two hidden layers than in 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 Fig. 1, use training sets of \( V \)-\( i \_ d_n \) input-output pairs for model adaptation (see next section), where \( V \) is the initial time steps interval employed for training purpose.

Once we have established all the influences on the output, at moment \( t \), we apply Principal Component Analysis (PCA) (Jackson, 1991) (Sousa et al., 2007) to reduce the dimensionality of the input space and to un-correlate the inputs. Before applying PCA, we had preprocessed the inputs and outputs, by using: replacement of missing values (NaN); detection and replacement (peak-shaving) of outliers; and, finally, normalization. Data preprocessing prepares raw data for the forecasting model and transforms it into a format that will be easier and more effectively processed. We have applied the reverse process of normalization, in order to denormalize the simulated outputs. Data preprocessing and data postprocessing are essential steps of the knowledge discovery process, in real world applications, and they greatly improve the network's ability to capture valuable information, if they are correctly carried out.

3. Method description

The feature of "universal functional approximator" (Hornik et al., 1989) adds the power and flexibility of 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 neural network forecasting applications.

As the 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) during the training process.

Next, the adaptivity of the result is performed (and improved), by applying the retraining technique (Nastac et al. 2007), in a special way. This technique is a mechanism for extracting practical information directly from the weights of a reference
ANN that had been already trained in a preliminary phase. The retraining procedure reduces the reference network weights (and biases) by a scaling factor \( \gamma, 0 < \gamma < 1 \). The reduced weights are used further as the initial weights of a new training sequence, with the expectation of a better accuracy. The data that we have used in our model consist of \( V \times i \times d_n \) input-output pairs during each training (or retraining) phase, where \( V \) is the initial time/spatial steps interval employed for training purpose. As the splitting criterion, we randomly choose approximately 85\% of the data \( (V \times i \times d_n) \) for training set, and the rest for validation. Furthermore, we imposed the supplementary condition:

\[
E_{\text{rel}} \leq \frac{6}{5} E_{\text{tr}}
\]  

(4)

to avoid a large difference between the error of the training set \( (E_{\text{tr}}) \) and the error of the validation set \( (E_{\text{rel}}) \). 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, even though there is a real and separate test set of \( T \) different timesteps (where \( T < T \)).

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

1. Firstly, we decided the proper number of hidden neurons for each hidden layer \( N_{h1} \) and \( N_{h2} \) by testing several pyramidal ANN architectures. We chose the best model with respect to the smallest error between the desired and the simulated outputs. This error \( (E_{\text{net}}) \) was calculated for \( V \times i \times d_n \) data that included both training and validation sets.

2. Secondly, we predicted \( T \) 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 step \( t \), the neural network used as input the estimated outputs (besides the real inputs) that had been calculated at 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 running the risk that each step increases the forecasting error. Alternatively, we can use Always Real Inputs (ARI) approach, which employed the real previous outputs and not estimated ones.

Then, we computed the error ERR (Nastac et al. 2007) that represents the accuracy of the approximation of the output data, within the forecasting horizon of \( T \) steps:

\[
\text{ERR} = \frac{100}{T} \sum_{p=1}^{T} \frac{|O_{\text{rkp}} - O_{\text{fkp}}|}{|O_{\text{rkp}}| \cdot \frac{T}{T + p}}
\]

(5)

where \( T \) = number of times/space steps, \( O_{\text{rkp}} \) = real output \( k \) at step \( p \), and \( O_{\text{fkp}} \) = forecasted output \( k \) at step \( p \).

3. Thirdly, we applied the retraining technique for a shifted interval of timesteps between \( (\text{Shift}+1) \) and \( (\text{Shift}+V) \), where \( \text{Shift} \leq T \). Here we used the ANN architecture that resulted at the end of the previous step. We applied this technique for each value of \( \gamma \ (\gamma = 0.1, 0.2, \ldots, 0.9) \), 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 \( T \) values of the outputs (during the interval of timesteps \( (\text{Shift}+V+1) - (\text{Shift}+V+T) \)), in the same sequential mode as in step 2 (IS or ARI).

5. We repeated \( L \) times the steps 3 and 4 on successive shifted intervals of \( V \) timesteps for retraining processes and \( T \) timesteps for sequential forecasting. Each time the intervals were ascendingly repositioned with \( \text{Shift} \) time/space steps.
The retraining technique allows us to continuously improve the model, at times, by using new (shifted) databases. For a single combination of the delay vectors, we obtained (and used) a model with its associated adaptive behavior. The next two sections show how the above-mentioned steps were applied for various combinations of delay vectors in two general cases: time series and spatial sequences.

4. Time Series experimental results

Two financial and one industrial example are provided in the following.

A. Exchange Rate Forecasting

A typical example for time series analysis is the exchange rate forecasting. The particular system, which resulted by using our approaches, describes the relationship between 33 (or 36) input variables and one output variable that model the EUR-ROL exchange rate. The raw data consist of more than 2500 rows (time steps) – one data row every day during 7 years. We performed the steps described in Section 3 for various combinations of delay vectors. In Table 1 we present the values of test error (ERR according to (5)) for iterative simulations of the output, computed at the end of first training and, then, after each successive retraining phase (L = 40), when using In_Del = [1 2 3 4 5 6 8 12] and Out_Del = [0 1 2 4].

<table>
<thead>
<tr>
<th>Training / retraining interval</th>
<th>ERR (33 inputs)</th>
<th>ERR (36 inputs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IS</td>
<td>ARI</td>
</tr>
<tr>
<td>First training</td>
<td>4.3185</td>
<td>4.2391</td>
</tr>
<tr>
<td>Retraining 1</td>
<td>0.5592</td>
<td>0.54745</td>
</tr>
<tr>
<td>Retraining 2</td>
<td>0.73979</td>
<td>0.7293</td>
</tr>
<tr>
<td>Retraining 3</td>
<td>0.9866</td>
<td>0.95797</td>
</tr>
<tr>
<td>Retraining 4</td>
<td>0.95671</td>
<td>0.92525</td>
</tr>
<tr>
<td>Retraining 36</td>
<td>0.68876</td>
<td>0.66023</td>
</tr>
<tr>
<td>Retraining 37</td>
<td>0.37173</td>
<td>0.34761</td>
</tr>
<tr>
<td>Retraining 38</td>
<td>0.19405</td>
<td>0.18702</td>
</tr>
<tr>
<td>Retraining 39</td>
<td>0.20866</td>
<td>0.20311</td>
</tr>
<tr>
<td>Retraining 40</td>
<td>0.38001</td>
<td>0.35892</td>
</tr>
</tbody>
</table>

We carried out the simulations under the following assumptions: \( V = 2200 \) days are enough for first training phase and then for each retraining phase; \( T = 1 \) (or 3, 7, 15, 30) days represent the prediction interval; and \( \text{Shift} = 1 \) day is the shifting time for the next retraining. It is worth to mention that the values of these parameters can be easily changed. 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.
In Table 1, it seems that the delay vectors have been well chosen, since, finally, there has been a decrease in the test error, when successive retraining phases were performed. The table includes two kinds of columns for 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 quite remarkable that the Iterative Simulation provides pretty close values (usually a bit higher, but the difference is not significant) of the ERR as compared to the situation when the system is always fed with real inputs. Practically, a long-term forecasting can be implemented using only the Iterative Simulation and the second approach remains a utopia in the case of exchange rate forecasting.

An example of the error trends is showed in Figure 2, which includes two graphs of the ERR: the left one shows the evolution of the ERR when Iterative Simulations (IS) are performed, and the right one when the system Always uses the Real Inputs (ARI). One may note that the abscissa represents the numbers of the successive retraining phases and the first value 0 is associated with the first training.

![Graphs showing ERR trend](image)

**Fig. 2.** ERR trend (Case 33 inputs) of test sets for the first training and L = 40 successive retraining phases.

Also, others values of previous parameters can be easily changed. The prediction interval (T) can be, for example, enlarged to 60 timesteps (two months) or more. The quality of the predictions is graphically analyzed, by enforcing a tube around the real outputs, given by a function like the one below:

\[ f(n) = A + n \cdot q. \]  

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 \( \text{output}(n) \pm f(n) \), represented with dotted lines in Figure 3 that shows the graphs of the EUR/ROL exchange rate for the test interval of retraining 29. 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 \( n = 1 \ldots 40 \)). The abscissa shows the number of days in the years when predictions are performed.
One may 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. In this way, both lines (thin and thick) start from approximately the same point. The difference between $\text{ERR}_{\text{IS}} = 0.26649$ and $\text{ERR}_{\text{ARI}} = 0.25757$ is not significant. These values were computed by using the relation (5) for all the data represented in Figure 3. It is noticeable that there are not visible differences of these two graphs. It seems that the model is robust and works very well with the iterative simulations, as well as on the ARI way. We obtained similar results in all situations without any exception.

![Iterative simulations of output and Always real inputs](image)

**Fig. 3.** Data forecasting for the test interval of retraining 29. $\text{ERR}_{\text{IS}} = 0.26649$ and $\text{ERR}_{\text{ARI}} = 0.25757$.

We noticed that, except for a few cases, in almost all the graph representations of the predictions the trends were well captured (even outside the “tube”) by using our approach. Moreover, we studied both graphs (with iterative simulation and when the system always uses the real inputs) in order to demonstrate the robustness of the model. The Iterative Simulation does not increase the error as much as one could expect at the first sight. 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. Nevertheless, we remark sometime interesting evolutions of the predictions that return, after a while, to the desired range.

### B. Stock Market Forecasting

The HEX Forest Industry Index is an important index in Finland economy (HEX - Helsinki Stock Exchange) that illustrates the average global trend in forest industry. The goal was to find a practical mathematical model that describes the relationship between 8 input variables (all of which have measurable influences), and one output variable that model a forecasting process of this HEX Forest Industry Index. All inputs and output vary dynamically, and different time-delays might occur there. Changing an input variable may result in an output change that starts only a day or a couple of days later and goes on for up to several weeks. The raw data consist of 2960 rows (timesteps) – one data row every working day during 11 years.

In Table 2 we present the values of test error, for the training and the successive retraining phases, when using: $\text{In}_{\text{Del}} = [5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 12 \ 16]$ and $\text{Out}_{\text{Del}} = [0 \ 1 \ 2 \ 4]$. 

84  
ICAOR 2010 Turku, Finland
Table 2. Evolution of test error (ERR)

<table>
<thead>
<tr>
<th>Retraining</th>
<th>Training or retraining interval</th>
<th>Test interval</th>
<th>Test Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>1 - 2920</td>
<td>2921 - 2940</td>
<td>4.2686</td>
</tr>
<tr>
<td>Retraining 1</td>
<td>2 - 2921</td>
<td>2922 - 2941</td>
<td>3.6703</td>
</tr>
<tr>
<td>Retraining 2</td>
<td>3 - 2922</td>
<td>2923 - 2942</td>
<td>2.9412</td>
</tr>
<tr>
<td>Retraining 3</td>
<td>4 - 2923</td>
<td>2924 - 2943</td>
<td>2.7541</td>
</tr>
<tr>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Retraining 20</td>
<td>21 - 2940</td>
<td>2941 - 2960</td>
<td>1.8278</td>
</tr>
</tbody>
</table>

We have selected these results out of five different combinations of delay vectors. It seems that the delay vectors have been well chosen, since, on average, there has been a decrease in the test error, when successive retrainings were performed.

![Graph](image)

Fig. 4. Data forecasting for test interval of retraining 20.

Figure 4 shows the graphs of HEX forest index for last test interval (of retraining 20). The real data are represented with thin lines and the neural network values with thick lines, also as in the previous example. We enforced a tube around the real outputs, given by the function f(n) = 100 + 0.05 \cdot n.

C. Industrial Forecasting

The goal is to find a practical mathematical model that describes the relationship between 16 input variables and 4 output variables that model a process in glass manufacturing. The raw data consist of 16000 rows (timesteps) – one data row every 15 minutes during 6 months. The experimental results lead to similar conclusions as previous two examples: a general decreasing, on average, of ERR when the delay vectors are well suited. It is possible to work with 4 different models (one model for each output) or a single model with 4 outputs. For this industrial example it seems that the outputs have different meanings and different time-delay behaviors. Consequently, the normal way is to split the initial system in four subsystems. Good results were obtained when we started to work under the following assumptions: V = 7500 timesteps data for first phase and then for each retraining phase; T = 500 timesteps represent the prediction interval; and Shift = 500 timesteps is the shifting time for the next retraining. It is not necessary that the last two factors have the same values. The prediction interval can be, for example, enlarged to 1500 timesteps (see Fig. 5). The number of samples for the training (or retraining) interval can be modified according to the experience accumulated. The enforced tube is given by the function f(n) = 3 + 0.003 \cdot n.
Furthermore the model seems to be quite robust. In many cases (like Fig. 5), the model does not increase the forecasting error when a reasonable number of iterative (sequential) forecast processes were encountered. This is a very positive effect even was not expected.

5. DNA Sequence Forecasting

In this case we used a modified architecture based on only one delay vector: Out Del (see Fig. 6). Therefore, the system (model) tries to match the current value of the output, by properly adjusting a function of its previous values from the DNA sequence. In Figure 6, by $t$ we mean the spatial position $t$ in a sequence.

We show in the following the results obtained when using prokaryote and eukaryote DNA genomic sequences. In contrast with time series approach, where the Iterative Simulation (IS) is viewed as the practical approach, whereas here a more realistic approach is when the system Always uses the Real Inputs (ARI). In this case, the real previous outputs and not the estimated ones are considered for each forecasting step.

A. Prokaryote DNA sequences forecasting
The prokaryote DNA sequence used in our experiments is a series of 100,000 values that represents a part of Escherichia coli DNA. This common bacterium has been studied intensively by geneticists because of its small genome size. In Table 3 we show the value of test error (ERR) for iterative simulations (IS) of the output and when the
system always uses the real inputs (ARI). The values of ERR were computed at the end of first training and, then, after each successive retraining phase ($L = 76$), when using: $Out_{Del}$= [4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64], $T$=5000 spatial steps, $T$=500 steps as prediction horizon, and Shift = 500 as shifting interval for the next retraining.

**Table 3.** Evolution of test error (ERR) for Escherichia coli DNA forecasting

<table>
<thead>
<tr>
<th>Retraining phase</th>
<th>Training/retaining interval</th>
<th>Test interval</th>
<th>Test Error (IS)</th>
<th>Test Error (ARI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>First training</td>
<td>1–5000</td>
<td>5001–5500</td>
<td>11.411</td>
<td>2.3012</td>
</tr>
<tr>
<td>Retraining 1</td>
<td>501–5500</td>
<td>5501–6000</td>
<td>1.9404</td>
<td>0.8820</td>
</tr>
<tr>
<td>Retraining 2</td>
<td>1001–6000</td>
<td>6001–6500</td>
<td>9.8763</td>
<td>2.3449</td>
</tr>
<tr>
<td>Retraining 3</td>
<td>1501–6500</td>
<td>6501–7000</td>
<td>3.6978</td>
<td>0.8211</td>
</tr>
<tr>
<td>Retraining 74</td>
<td>37001–42000</td>
<td>42001–47000</td>
<td>1.2897</td>
<td>0.17906</td>
</tr>
<tr>
<td>Retraining 75</td>
<td>37501–42500</td>
<td>42501–47500</td>
<td>1.1096</td>
<td>0.09026</td>
</tr>
<tr>
<td>Retraining 76</td>
<td>38001–43000</td>
<td>43001–48000</td>
<td>0.4779</td>
<td>0.08671</td>
</tr>
</tbody>
</table>

In Table 3, there is a global decreasing trend of these errors computed for 500 steps. There is a significant difference between ERR$_{IS}$ and ERR$_{ARI}$ (see Fig. 7). This time the abscissa represents the first training (at 0) and then the successive values of retraining phases.

If we enlarge the prediction horizon to $T$=5000 steps we can visual estimate when the predicted data deviate from the real ones (Fig. 8). The tube around the real outputs is given by the function $f(n)$=1+0.05·$n$.

In Fig. 7 we can easily consider that only first 2000 steps proved an accurate estimation of the output when the system always uses the real inputs. Clearly, the iterative simulation is not as precise as the second one (for Always real inputs).

When $T$=5000, the decreasing trend of ERR is not clearly stated. This means that we must impose a limit on the prediction interval. The system has to be retrained after maximum 2000 steps (see Fig. 8) in order to obtain a reasonable prediction.

**Fig. 7.** ERR trend of test sets (with $T$=500 steps) for first training and $L$=76 successive retraining phases.
Fig. 8. Data forecasting for test interval of retraining 75 when test set is enlarged to 5000 steps. ERRARI = 1.4388.

We performed different experiments changing the values of Out_Del and the results are rather similar.

B. Eukaryote DNA sequences forecasting
We employed a DNA sequence from the human chromosome 22 in order to observe the eukaryote DNA sequences forecasting capability of our model. Maintaining the same parameters as in the previous experiments, but using Out_Del=[1, 2, 3, ..., 35] with 35 successive values, we finally obtained similar results. It seems that the general algorithm improves the forecasting results irrespective of the genomic sequence kind, having a large tolerance for the vector Out_Del.

6. Conclusions
The paper presents an adaptive predictive system that can be used for non-stationary sequences in time or space domains. It is crucial to start with an optimal system in what concerns the architecture and the choice of delay vectors. This is the reason that our approach must take a relatively long time in the first phase, for searching among many possible solutions. The updating mechanism is the adaptive retraining technique, which has to be performed periodically (at intervals that depend on the application). The SCG algorithm has been used for (re)training the ANN, even if it is not the fastest one. Its great advantage is that it works very efficiently for networks with a large number of weights, does not require large computational memory, has a good convergence and is very robust. Furthermore, as we always use the early stopping method (validation stop) during the training, it is better to avoid algorithms that converge too rapidly, such as Levenberg-Marquardt (LM). The SCG is well suited for 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. 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 both the interpolation among the a priori defined points, as well as the extrapolation outside the range bordered by the extreme points of the training set.

The evaluation of test error shows that the adaptive retraining technique can gradually improve, on average, the achieved results. 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. 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 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. This means that the old information accumulated during the older trainings will be slowly forgotten and the learning process will be adapted to the newest evolutions of the process, irrespective of its kind: temporal or spatial. A large variety of forecasting applications (as it was shown in this paper) for nonstationary sequences can be solved using the adaptive retraining technique.

Current research targets the implementation of this neuro-adaptive system for climate forecasting using the North Atlantic Oscillation influence in an European region.

References