AN EVOLUTIONARY AND CONNECTIONIST APPROACH FOR TIME SERIES FORECASTING

Paulo Cortez, Miguel Rocha and José Neves
Universidade do Minho, Portugal

Abstract

The combination of the evolutionary and connectionist paradigms for problem solving takes a strong inspiration from living systems and is gaining an increasing attention when it comes to the development of computational systems that can handle complex and dynamic problems.

One’s claim is that Time Series Forecasting is a fertile domain for the test of these technologies. Therefore, a number of experiments were conducted in order to evaluate the merits or demerits of the approach, being the results compared with those obtained from the use of conventional procedures (e.g., the Holt-Winters and the ARIMA ones).

Introduction

Nowadays, the fierce competition between individuals and organizations is a trademark of modern societies, where the gain of strategic advantages may be the key to success. Therefore, the ability to forecast the future, based on past data, makes a leverage that can push the organizations or individuals towards windows of opportunity. Time Series Forecasting (TSF), the forecast of a time ordered variable, is an important tool in this scenario, since it makes possible to predict the behaviour of complex systems, solely by looking at data patterns in past data, thus not requiring the presence of all the parties involved. Conventional TSF, coming from disciplines as Operations Research and Control Theory may provide good forecasts when linear data is involved [6]. However, when a higher degree of non-linearity is presented these traditional approaches may not be the most adequate [9].

Several recent breakthroughs in the nouvelle Artificial Intelligence (AI) area have been carried out by designing new optimization procedures, based on analogies with the evolution of natural living systems. Techniques such as the Artificial Neural Networks (ANNs) and Genetic and Evolutionary Algorithms (GEAs) have already on their shoulders some good results on a broad set of scientific and engineering problems, such as the ones of Combinatorial and Numerical Optimization, Pattern Recognition, Computer Vision or Robotics. In this work one develops an approach to TSF based on an architecture that aims to achieve the best of the two worlds.

Artificial Neural Networks in TSF

Time Series (TS) often present characteristics of significant noise components and non-linearity, making it a well suited domain for the application of ANNs. The idea is to train the ANN with past

---

1 This work was supported by FCT through the PRAXIS’ grant XXI/BD/13793/97
data, and then use this trained network to predict future values. In the process of training, the ANN should incorporate the main patterns present in the data. In this context, there are two main candidates, Recurrent Networks or Time Lagged Feedforward Networks (TLFNs) [4]. The use of TLFNs for TSF began in the late eighties, with encouraging results, namely when applied to financial markets, and the field has been consistently growing since [9][2].

A TLFN is composed of a tapped delay line memory, the dynamic part that accounts for time, and a static feedforward network, which handles nonlinearity. TLFNs perform one step ahead forecasts by using a sliding time window of \( n \) lags from the TS, for an ANN with \( n \) input nodes (Figure 1). One has the output (or forecasted value) as a function of \( n \) previous ones. If one takes the function implemented by the ANN to be \( f() \), the value of the forecast at time \( t \) is given by \( F_t = f(x_{t-n}, \ldots, x_{t-1}) \), where \( x_1, x_2, \ldots, x_t \) denotes the TS.

![Figure 1: Structure of the TLFN](image)

In the task of selecting the best ANN’s topology for a given TS, two issues must be contemplated: how to define the inner structure of the ANN; i.e., its hidden layers, number of neurons per layer, and connections (or synapses) among neurons; and how to set the connection’s strengths (or weights); i.e., the extension by which a signal is amplified or diminished by a connection, as given by some training procedure.

The logistic (or sigmoid) activation function will be used on the hidden nodes, to enhance nonlinearity, and the linear activation function will be used on the output node, to scale the range of the output, since the codomain of the logistic function is \( [0, 1] \).

The next level of specialization is concerned with the selection and preprocessing of data from a TS, in order to obtain training cases, making possible that repeated training epochs successively improve the performance of the network. In each step, the training data is obtained by data filtering; i.e., the selection of the relevant patterns to feed the ANN, which is closely related to the way the TS’ sliding window is built.

**GEANN Systems**

It is common to address each of these questions with trial-and-error heuristics or gradient descent based procedures, which tend to be unsuitable due to the huge size of the search spaces involved. Under these conditions, the use of Genetic and Evolutionary Algorithms (GEAs) can be a better alternative. GEAs are
general purpose problem solving tools, based on mechanisms abstracted from population genetics [3]. A GEA maintains a set of trial solutions, a population, and operates in cycles named generations, that produce successive populations by survival-of-the-fittest selection, followed by genetic recombination. Trial solutions are represented as strings called chromosomes, being crossover and mutation the most commonly employed operators. Crossover produces offspring (i.e., new trial solutions) by recombining the information from two or more ancestors, while mutation aims to prevent unexpected convergence to local optima.

One may combine GEAs and ANNs in different ways. In the past, combinations have been both supportive (i.e., they have been used sequentially), and collaborative (i.e., the have been used simultaneously). The former ones work by preparing data for consumption by the other, namely using a GEA to select features to be used by ANNs’ classifiers. Collaborative combinations, on the other hand, use GEAs to evaluate the ANN’s connection strengths or weights, the ANNs topology, or both [10].

The populations considered in the GEA are made of a pool of individuals, each one coding a different ANN, which will be evaluated according to its forecasting accuracy (or fitness), typically measured by the error returned from the training procedure. The Resilient Back-Propagation (RPROP) algorithm, a more efficient variant of the well known back-propagation algorithm, was adopted for the training, being the initial weights randomly generated within the range \([-\frac{2}{\sqrt{i}}, \frac{2}{\sqrt{i}}]\), for a node with \(i\) inputs [8].

Estimating the accuracy of an ANN is important, not only to predict its forecasting capability, but also for model selection; i.e., to choose the best ANN’s topology. Here, the main concern is related to the overfitting phenomenon; i.e., after an unknown number of learning epochs, the ANN loses generalization capabilities, focusing on the particularities of the given cases. One way to prevent this is to adopt early stopping, where the learning procedure is stopped after the growth of some estimation error metric [4]. This estimation can be provided by cross-validation, a standard statistic tool, where the training cases are split into two sets: a training set (to assimilate the patterns), and a validation set (to test the ANN’s generalization capability). A more effective variant is \(K\)-fold cross validation, where the training examples are divided into \(K\) subsets, \(K > 1\). The learning model is trained on all subsets except for one, being the validation error measured on the subset left out. This procedure is repeated for \(K\) trials, each time using a different subset for validation. Finally, the performance of the model is achieved by averaging all validation errors.

**The Proposed Architecture**

In this work one proposes an architecture for TSF based on a GEANN system. The main idea is to use GEAs in two different tasks: selecting the best sliding window for a given TS; i.e., choosing the time lags that provide useful information in the forecast of a given value; and selecting the ANN’s topology for a given TS; i.e., choosing which connections should be used among the nodes.

In the previous tasks one uses GEAs with a binary representation. In the first case each gene codes a time lag, so if the \(i\)-th gene is 1, then the value \(x_{i-1}\) is used when forecasting \(F_i\). The chromosome will have 14 genes (time lags), since there is some evidence that this is the correct range, and a higher one would increase exponentially the searching time [2]. In the latter, each gene represents a possible connection of the ANN’s topology. If its value is 1 then the corresponding connection exists, otherwise it is not considered, except for the case of the connections between the hidden nodes and the output one, which will always exist, since this strategy enhances the creation of valid networks. One assumes a base network structure with \(n\) input and hidden nodes and \(n\) shortcuts, for a time window of \(n\) elements. Thus, the size of the chromosome will be given by \(n \times n + n\).
The overall architecture is presented in Figure 2. When a TS is presented to the system to be forecasted, the first task is to select the appropriate sliding time window, to create the ANN’s training cases. As referred to above, a GEA is used to achieve this task, by evolving a population of possible solutions (time windows), evaluating them (by training ANNs with the cases induced by the time windows and measuring the forecasting error on the validation examples as given by 10-fold cross-validation), selecting the most appropriate one. In this stage the ANNs are fully connected, with shortcut (or direct) and bias connections (Figure 1). Once the time window is selected and the training cases are created, the following task is to select the best ANN topology for the TS. Several topologies (solutions) are tested by the GEA, within a searching space that ranges from the simplest ANN, with only direct connections (without hidden nodes), to the fully connected network.

Results

All tests were performed under the Linux operative system, using an object-oriented software package, developed in C++, that allows the modular design of evolutionary applications [7]. There was some care in selecting TS that were related with real problems, from different domains such as financial markets or natural processes (Figure 3) [5].

Figure 3: The six series of Table 1
Table 1: Results

<table>
<thead>
<tr>
<th>Series</th>
<th>Window</th>
<th>GEANN</th>
<th>HW</th>
<th>ARIMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>bricks</td>
<td>&lt; 1...5, 8, 9, 12, 13&gt;</td>
<td>32.61</td>
<td>37.41</td>
<td>-</td>
</tr>
<tr>
<td>chemical</td>
<td>&lt; 1, 2, 4, 10, 12, 13&gt;</td>
<td>0.35</td>
<td>1.61</td>
<td>0.47</td>
</tr>
<tr>
<td>passengers</td>
<td>&lt; 1, 4...12, 14&gt;</td>
<td>21.42</td>
<td>15.87</td>
<td>20.08</td>
</tr>
<tr>
<td>prices</td>
<td>&lt; 1 &gt;</td>
<td>7.53</td>
<td>10.72</td>
<td>7.66</td>
</tr>
<tr>
<td>sunspots</td>
<td>&lt; 1...5, 7, 9...14&gt;</td>
<td>17.27</td>
<td>62.8</td>
<td>21.31</td>
</tr>
<tr>
<td>traffic</td>
<td>&lt; 1...5, 8, 11, 12&gt;</td>
<td>21.51</td>
<td>20.97</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 4: GEANN’s forecasts for series sunspots and the topology of the best ANN (shortcuts omitted)

Table 1 shows the system’s results for each series. The structure of the time window is given in the form < 1, 2, ..., n − 1, n >, where each number represents the correspondent time lag. The results show that the GEANN tends to filter few lag values, since time window sizes range from 8 to 12. The exception occurs in series prices due to its strong trend component. In fact, the ARIMA model for this series uses only the same time lag. The last three columns show the error, in terms of the Root Mean Squared Error (RMSE), obtained when forecasting the last 10% values with the GEANN system and other forecasting methodologies (Holt-Winters [6] and ARIMA [1]). The results are encouraging since the conventional forecasting techniques are outperformed on 4 of the series. In fact, the ARIMA results are only better than the GEANN ones for series passengers, which is a seasonal one. The GEANN system seems to present more difficulties with this kind of series (bricks, passengers and traffic), since the Holt-Winters results are better for the last 2 ones. This result is not surprising, since the Holt-Winters methodology was developed specifically to tackle this particular type of TS.

As an example, one will consider series sunspots (Figure 4). As expected, the GEANN’s system has a good performance on this series, since the forecasts rely close to the real values. The best topology has 64 connections less than the based fully connected one, which means a 41% connection’s reduction.

Conclusions

New exciting possibilities have been created with the surge of new optimization techniques, like
ANNs and GEAs. More recently, the research community started to focus on the possibility to combine both, in order to take better advantage of their potentialities [10]. This work contributes to preempt the lack of real world GEANN’s applications. In particular, results show that TSF is a promising field for the use of GEANNs systems, specially for TS with a high degree of nonlinearity (such as series sunspots or chemical). Comparative results show that the GEANN’s systems can perform better than conventional methodologies (such as ARIMA and Holt-Winters). However, for series with a high seasonal component, the Holt-Winters approach continues to be more appealing. The main drawback of the proposed system is the computational effort that is required in order to get the best forecasting model. On the other hand, the system works in a automatically way, with a minimum of human intervention, contrary to ARIMA, which requires the presence of experts.

In future work one intends to improve the system’s robustness; explore the use of other topologies, like Recurrent ANNs [4]; and examine other gene codification schemes such as the cellular one [10].

References


