Applied Soft Computing 23 (2014) 432–443

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Evolutionary optimization of sparsely connected and time-lagged neural networks for time series forecasting

Juan Peralta Donate[a], Paulo Cortez[b,*]

[a] Universidad Autónoma de Barcelona, Edifici O, Campus UAB, Bellaterra, 08193 Barcelona, Spain
[b] ALGORITMI Research Centre, Department of Information Systems, University of Minho, Campus de Azurém, 4800-058 Guimarães, Portugal

A R T I C L E I N F O

Article history:
Received 31 October 2012
Received in revised form 24 June 2014
Accepted 24 June 2014
Available online 2 July 2014

Keywords:
Estimation distribution algorithm
Multilayer perceptron
Time series
Regression

A B S T R A C T

Time series forecasting (TSF) is an important tool to support decision making (e.g., planning production resources). Artificial neural networks (ANNs) are innate candidates for TSF due to advantages such as nonlinear learning and noise tolerance. However, the search for the best model is a complex task that highly affects the forecasting performance. In this work, we propose two novel evolutionary artificial neural networks (EANNs) approaches for TSF based on an estimation distribution algorithm (EDA) search engine. The first new approach consist of sparsely connected evolutionary ANN (SEANN), which evolves more flexible ANN structures to perform multi-step ahead forecasts. The second one, consists of an automatic Time lag feature selection EANN (TEANN) approach that evolves not only ANN parameters (e.g., input and hidden nodes, training parameters) but also which set of time lags are fed into the forecasting model. Several experiments were held, using a set of six time series, from different real-world domains. Also, two error metrics (i.e., mean squared error and symmetric mean absolute percentage error) were analyzed. The two EANN approaches were compared against a base EANN (with no ANN structure or time lag optimization) and four other methods (autoregressive integrated moving average method, random forest, echo state network and support vector machine). Overall, the proposed SEANN and TEANN methods obtained the best forecasting results. Moreover, they favor simpler neural network models, thus requiring less computational effort when compared with the base EANN.

Introduction

Nowadays, forecasting the future using past data is an important tool to reduce uncertainty and support both individual and organization decision making. For example, multi-step predictions (e.g., issued several months in advance) are useful to aid tactical decisions, such as planning production resources. In particular, the field of time series forecasting (TSF) deals with the prediction of a given phenomenon (e.g., ice cream sales) based on the past patterns of the same event. TSF has become increasingly used in areas such as agriculture, finance, management, production or sales.

Several Operational Research TSF methods have been proposed, such as Holt-Winters (in the sixties) or the Autoregressive Integrated Moving Average (ARIMA) methodology [30] (in the seventies). More recently, several Soft Computing methods have been applied to TSF, such as Artificial neural networks (ANN) [13], evolutionary computation (EC) [10] and fuzzy techniques [27] Also, several hybrid systems that combine two or more soft computing and/or forecasting techniques have been proposed for TSF, such as proposed in [2,23,24,35].

This paper is focused on the use of ANN [20], which are a natural solution for TSF due to advantages such as flexibility (i.e., no *a priori* knowledge is required), nonlinear learning and robustness to noisy data. ANN were initially applied to TSF in 1987 [25] and such research has been consistently growing since [13,33,44]. Some examples of successful ANN forecasting applications are Internet traffic [9], air pollution [34] and financial markets [24].

While several types of ANN have been proposed for TSF (e.g., radial-basis functions, recurrent networks), the majority of the studies adopt the multilayer perceptron architecture [13,25,44]. In particular, the time-lagged feedforward neural network (TLFN) is a popular approach [9,20,33]. The TLFN adopts a multilayer perceptron ANN as the learning base model and uses a sliding time window method to create supervised training examples. The sliding time window defines a set of time lags that are used as inputs by the ANN.

* Corresponding author. Tel.: +351 253510313; fax: +351 25351031300.
E-mail addresses: jperalta@cvc.uab.es (J. Peralta Donate), pcortez@dsi.uminho.pt (P. Cortez).

http://dx.doi.org/10.1016/j.asoc.2014.06.041
1568-4946/© 2014 Elsevier B.V. All rights reserved.

When adopting multilayer perceptrons for TSF (i.e., TLFN), a crucial issue is the design of the best forecasting model, which involves both feature and model selection [13,40]. The former is required since a small set of time lags will provide insufficient information to the ANN, while using a high number of time lags will increase noise and probability of having irrelevant inputs. Indeed, time lag selection is a core step of the ARIMA methodology, which often selects the 1, 12 and 13 time lags for monthly seasonal and trended series [30]. The latter selection is needed to get a good generalization capacity, since a too complex ANN model will overfit the data, while a model that is too simple will present limited learning capabilities. However, most ANN works for TSF adopt a manual design for this feature and model selection that is *ad hoc* (e.g., [9,12,22,25,33,44]), based either in domain knowledge or in trial and error experimentation. An alternative is use Evolutionary Computation to search for the best ANN, in what is known as Evolutionary ANNs (EANNs) [15,38,42]. Often, EANNs require more computation when compared with manual ANN design, since more ANNs are tested. Yet, EANNs are much more appealing to non specialized users, given that few parameters need to be selected, the search is fully automatic and more exhaustive, thus tending to provide better performances when compared with the manual design.

EANN systems have been treated mainly using three different optimization points of view [6,42]: topology (e.g., number of hidden layers, number of nodes in each layer); connection weights (e.g., values for each ANN connection); and learning rules (e.g., learning factor). Within the TSF domain, the majority of EANN works make use of rather rigid ANN structures that are fully connected, evolving only ANN hyperparameters, such as number of input and hidden nodes [6,34]. For instance, once the number of inputs is set, all time lags are adopted by the TLFN. Working with fully connected structures also means that ANNs can be more complex than needed. As a consequence, these EANNs tend to require an heavy computational effort. Moreover, most EANN works for TSF use the standard Genetic Algorithm (GA) as the search engine, which requires setting several parameters to (e.g., mutation rate, population size). The Estimation Distribution Algorithm (EDA) is a more recent Evolutionary Computation variant, proposed in 2001 [26], and that makes use of exploitation and exploration properties to find good solutions. When compared with other search methods (e.g., GA), EDA has the advantage of requiring just one parameter (i.e., population size), since crossover and mutation processes do not exist in EDA. Also, EDA has a fast convergence and in recent previous work [35] it has outperformed the standard GA and differential evolution methods when selecting the best ANN TSF models.

In this paper, we propose two novel EANN variants for TSF that are fully automatic and can be used by non specialized users to perform multi-step ahead time series forecasts, since no *a priori* knowledge is assumed from the analyzed time series. In contrast with the majority of EANN works for TSF, the proposed EANN variants make use of EDA as the search engine under two design strategies: Sparsely connected EANN (SEANN) and Time lag selection EANN (TEANN). Both strategies perform a simultaneous feature and model selection for TSF, although with a different emphasis. SEANN puts more effort in model selection by explicitly defining if a connection exists and time lag deletion only occurs when an input has no connections. TEANN enforces feature selection, explicitly defining which time lags are used in the chromosome, while ANN structure selection is made only in terms of number of input and hidden nodes. These strategies are addressed separately in order to measure the contribution of each other when compared with the fully connected EDA EANN [35]. Moreover, we also compare all EANN methods with the popular ARIMA methodology and three recently proposed machine learning methods: Random Forest (RF), Echo State Network (ESN) and

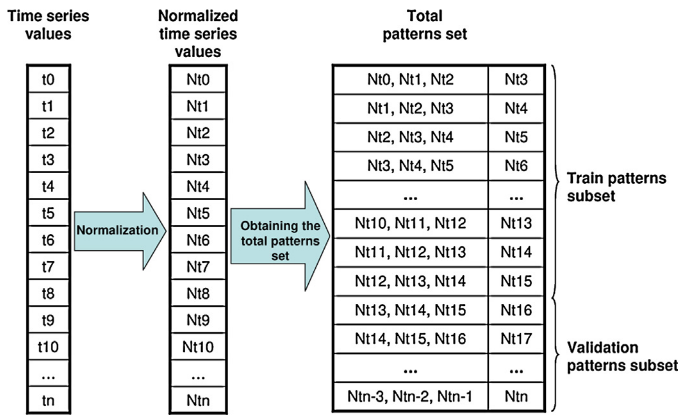

Fig. 1. Process to obtain training and validation sets.

Support Vector Machine (SVM). The experiments were performed using several real-world time series from distinct domains and the distinct forecasting approaches were compared under both forecasting and computational performance measurements. The paper is organized as follows. First, section "Evolutionary design of artificial neural networks" described the EANN approaches. Next, in section "Experimental setup and results" we present the experimental setup and analyze the obtained results. Finally, we conclude the paper in section "Conclusions".

Evolutionary design of artificial neural networks

Time series and ANN

The problem of forecasting time series with ANN [35] is considered as obtaining the relationship from the value at period y_t (in this system the resulting ANN will have only one output neuron) and the values from previous elements of the time series, using several time lags $\{t-1, t-2, \ldots, t-I\}$, to obtain a function:

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-I}) \quad (1)$$

where $\hat{y}_t$ denotes the estimated forecast, as given by the ANN (f), and I the number of ANN input nodes.

In order to obtain a single ANN to forecast time series values, an initial step has to be done with the original values of the time series, i.e., normalizing the data. The original values (y_t) are normalized into the range $[0, 1]$ (leading to the N_t values). Once the ANN outputs the resulting values, the inverse process is carried out, rescaling them back to the original scale. Only one neuron was chosen at the output layer and multi-step ahead forecasts are built by iteratively using 1-ahead predictions as inputs [9]. Therefore, the time series is transformed into a patterns set depending on the k inputs nodes of a particular ANN, each pattern consisting of:

- I inputs values, that correspond to I normalized previous values: $N_{t-1}, N_{t-2}, \ldots, N_{t-I}$.
- One output value: N_t (the desired target).

This patterns set will be used to train and validate (i.e., compute fitness value) each ANN generated during the evolutionary execution. Thus, the patterns set is split into two subsets, using a timely ordered holdout scheme with 70% of the elements for training and the most recent 30% elements for validation. We note that the 70/30 split is very common (e.g., [22,27]) and in [12] this split provided better TSF results for ANN when compared with other divisions (e.g., 60/40 and 80/20). As an example, Fig. 1 shows how such training and validation sets are created with $I = 3$. Finally, after evolving the ANN, the best model is evaluated on a test set, which