View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Universidade do Minho: RepositoriUM

Evolutionary Optimization of Sparsely Connected and Time-Lagged Neural Networks for Time Series Forecasting

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

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

Abstract

Time Series Forecasting (TSF) is an important tool to support decision making (e.g., planning production resources). Artificial Neural Networks (ANN) 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 (EANN) 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

Email addresses: `jperalta@cvc.uab.es` (Juan Peralta Donate), `pcortez@dsi.uminho.pt` (Paulo Cortez)

results. Moreover, they favor simpler neural network models, thus requiring less computational effort when compared with the base EANN.

Keywords: Estimation Distribution Algorithm, Multilayer Perceptron, Time Series, Regression,

1. 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 [23, 2, 35, 24].

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 [44, 33, 13]. 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 [25, 44, 13]. In particular, the Time-Lagged Feedforward Neural Network (TLFN) is a popular approach [20, 33, 9]. 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.

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., [25, 44, 33, 12, 9, 22]), 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) [42, 38, 15]. 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 [42, 6]: 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 [34, 6]. 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 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 2 described the EANN approaches. Next, in Section 3 we present the experimental setup and analyze the obtained results. Finally, we conclude the paper in Section 4.

2. Evolutionary Design of Artificial Neural Networks

2.1. 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:

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

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 includes the most recent y_t elements that were not used during the EANN procedure.

2.2. Evolutionary Neural Network Design

The problem of designing ANN could be seen as a search problem into the space of all possible ANN. While several EC methods could be used for this search, we adopt in this paper the EDA algorithm, since it has outperformed the standard GA in our previous work [35]. As a base ANN structure, we adopt the TLFN with all time lags, from 1 to i, as the forecasting model. We use fully connected multilayer perceptrons with only one hidden layer and one output node [33, 35].

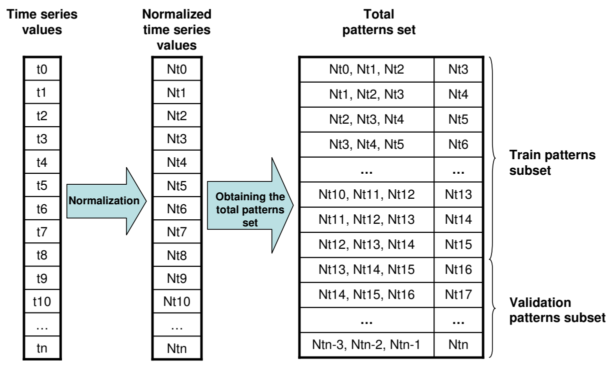


Figure 1: Process to obtain training and validation sets.

The Resilient Propagation (RPROP) learning algorithm is an enhanced version of the backpropagation algorithm, performing a local adaptation of the weight-updates based on behavior of the error function, given by the local gradient information. When compared with other algorithms (e.g., backpropagation), the RPROP presents a faster training, requiring less computational effort [37, 31].

In this paper, an evolving hybrid system that uses EDA and RPROP learning algorithm, is adopted. This approach uses a digit number representation (i.e., $\in \{0, ..., 9\}$) to encode the ANN topology hyperparameters and RPROP learning parameters, with multiple initializations. Under this EANN, the hyperparameters that we optimize are the number of inputs nodes (i) and number of hidden neurons (h) of the hidden layer. For the ANN learning, we use the RPROP algorithm [37], which presents a faster training convergence when compared with other algorithms (e.g., Backpropagation), requiring less computational effort. The performance of RPROP may depend on the correct adjustment of two numeric parameters, known as $\Delta_{max} \in \Re$ (although the default value is 50.0) and $\alpha \in [0, 1]$ (although the value is typically closer to 0), also known as Δ_0. Hence, we adopt a direct encoding schema, which places into the chromosome (Fig. 2): two decimal digits (i.e., from 0 to 9), to codify the number of inputs nodes (i); another two digits for the number of hidden nodes (h); two more $\Delta_{max} \in 0, 1, \ldots, 99$ and finally one gene for $\alpha = \{1, 0.01, 0.001, \ldots, 10^{-9}\}$.

As explained in Section 1, EDA contains just one parameter, the popula-

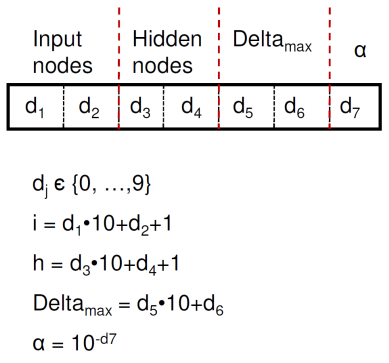

Input nodes
Hidden nodes
Delta$_{max}$
α
d_1 d_2 d_3 d_4 d_5 d_6 d_7
$d_j \in \{0, \ldots, 9\}$
$i = d_1 \cdot 10 + d_2 + 1$
$h = d_3 \cdot 10 + d_4 + 1$
$Delta_{max} = d_5 \cdot 10 + d_6$
$\alpha = 10^{-d7}$

Figure 2: EANN encoding scheme.

tion size (P). Since the EDA works as a second order optimization procedure, the tuning of this internal parameter is not a critical issue. In this work, a fixed value of $P = 50$ was adopted. In preliminary experiments, we tested a sensitivity analysis with other values (e.g., $P = 48$ and $P = 52$) and achieved similar results.

The EANN search process consists of the following steps (Fig. 3):

1. A randomly generated population, i.e., a set of randomly generated chromosomes, is obtained.

2. The phenotypes (ANN architectures) and fitness value of each individual of the actual generation is obtained. To obtain the phenotype associated to a chromosome and its fitness value:

   (a) The phenotype of an individual of the actual generation is first obtained (using the Stuttgart Neural Network Simulator (SNNS) tool [43]).

   (b) Then, for each ANN, training and validation pattern subsets are obtained from time series data depending on the number of inputs nodes, as it was explained in Section 3.1.

   (c) The ANN is trained with RPROP (using SNNS). The architecture (topology and weights) of the ANN when the validation error (i.e., error for validation patterns subset) is minimum during the training process is stored (i.e., we adopt early stopping). Thus, this architecture is the final phenotype of the individual. The fitness for each individual is the minimum Mean Squared Error (MSE) validation error (Eq. 2), during the learning process.

3. Once the fitness values for whole population have been already obtained, Univariate Marginal Distribution Algorithm (UMDA)-EDA (with no dependencies between variables) [35] operators are applied in order to generate the next population. The UMDA operators work as follows. First, a truncation selection is adopted, which selects half of the best solutions from the current population. This subset population is included in the next population. Then, a distribution probability for the subset population is automatically estimated. For instance, if there are 25 individuals in the subset and only 5 of these individuals contain the first gene with a 1 digit, then the probability for setting this gene as 1 is set to $5/25 = 0.2$. New individuals are then sampled using the probabilities previously calculated and included in the next population.

4. Steps 2 and 3 are iteratively executed until a maximum number of generations is reached.

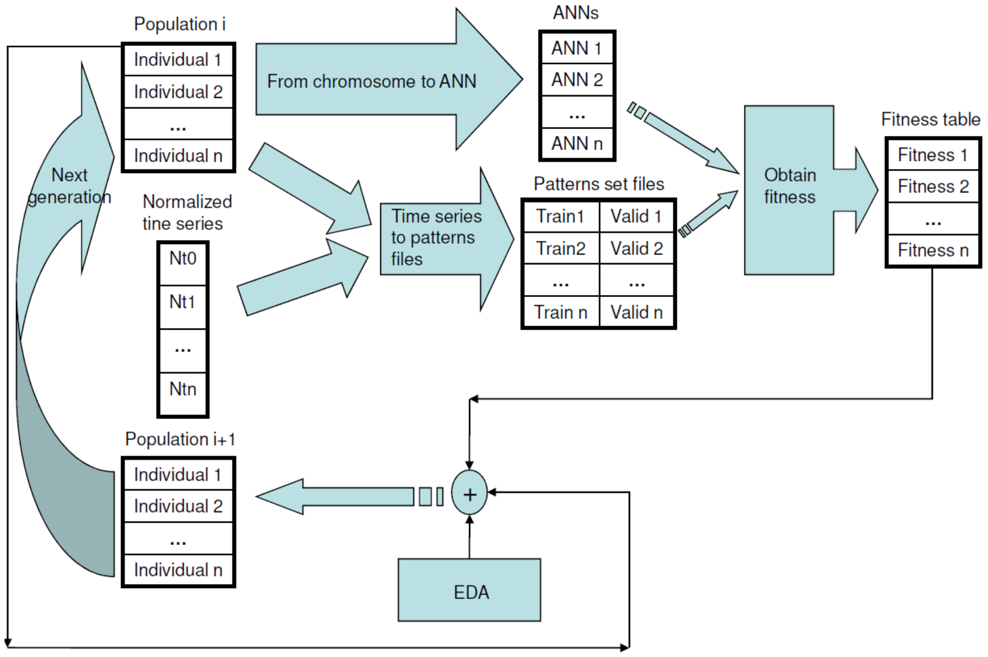


Figure 3: Schema of the EANN design.

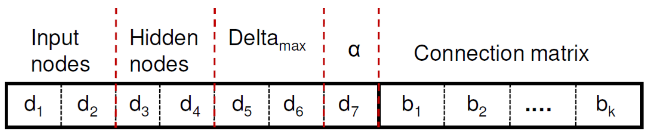
The fitness function uses the MSE and not the Mean Absolute Error (MAE) due to three main reasons. First, squared error metrics are more popular to validate forecasting methods [12, 30, 1]. Second, MSE is more sensitive to extreme errors (i.e., outliers) than MAE. Given that we use feedback of 1-ahead predictions to generate ANN multi-step forecasts, such outliers would dramatically propagate and affect the performance of long term predictions. Third, preliminary experiments with two series (Passengers and Temperature) have revealed better results for the MSE fitness approach when compared with the MAE one. In these preliminary experiments, the training data (e.g., 125 samples for Passengers) was further split into training (e.g., 74 samples for Passengers, used to train the ANN), validation (e.g., 32 samples for Passengers, used to compute the fitness) and test (e.g., last 19 of the 125 Passengers samples, used to compare the forecasting performance of MSE and MAE fitness approaches).

It should be noted that this EANN is competitive. This method was ranked at 6th position, when comparing the other Soft Computing methods, at the *NN5* competition [11]. EANN presented an average Symmetric Mean Absolute Percentage Error (SMAPE) error of 21.9% and also outperformed the ARIMA forecasts performed by the commercial Autobox tool (www.autobox.com, average SMAPE of 23.9%).

2.3. Sparsely Connected Evolutionary Artificial Neural Network

The topic of ANN topology selection was first suggested by Miller et al. [32], which proposed GA as a very good candidate for the search. Miller et al. identified two approaches to code the topology in a string: the strong specification scheme (or direct encoding scheme), where each connection of the network is specified by its binary representation, and a weak specification scheme (or indirect encoding scheme), where the exact connectivity pattern is not explicitly represented but it is computed on the basis of the information encoded in the string by a suitable developmental rule. Several authors have followed Miller et al. suggestion, including Whitley et al. [41] and Schaffer et al. [39], which adopted a direct encoding scheme. The main advantage of this direct approach is that it is easy to evolve networks with special connectivity properties, either by constraining the representations allowed or by including some specific penalty term in the fitness function. On the other hand, the disadvantage is that these representations do not scale well to large networks, with hundreds of nodes and thousands of connections. However, when dealing with networks of smaller size, the direct encoding scheme induces a less-fuzzy fitness function, i.e., the fitness value associated with each chromosome is more coherent when it is computed several times using different training sets.

The proposed Sparsely EANN (SEANN) works as the previous EANN (Section 2.2), except that now we can evolve more flexible structures. To achieve this, and following the work of [41] and [39], we adopted a direct binary encoding scheme that defines which connections are used by the ANN. The proposed chromosome includes three components. The first two components work exactly as described in Section 2.2. The maximum number of input and hidden nodes are needed to set the connection matrix size and the two RPROP parameters are used to train the ANN. The third component includes the direct binary encoding of the ANN connections. Fig. 4 shows the sparsely connected chromosome, where the last binary digits (i.e., connection matrix) set the active connections of the model. In Fig. 5, it can be


Input nodes
Hidden nodes
Delta$_{max}$
α
Connection matrix
d_1 d_2 d_3 d_4 d_5 d_6 d_7 b_1 b_2 .... b_k

$d_j \in \{0, \ldots, 9\}$, $b_j \in \{0,1\}$

$i = d_1 \cdot 10 + d_2 + 1$

$h = d_3 \cdot 10 + d_4 + 1$

$\mathrm{Delta}_{max} = d_5 \cdot 10 + d_6$

$\alpha = 10^{-d_7}$

Connection matrix $= b_1 \ldots b_k$

Figure 4: SEANN encoding schema (with the connection matrix).

observed an example of how the direct binary codification works, in order to obtain the ANN connection matrix from the chromosome. In general, each matrix cell represents a valid connection between an incoming node (at the row) with the outgoing node (at the column). In the example, the third digit ($b_3 = 1$) sets a connection between the first input node and the third hidden node. The exception is the last row, which represents the connections between the hidden nodes (at the columns) and the output one (the last row). By default, the largest possible connection matrix is always set, with the dimensions $Row \times Col$, although the real dimensions are limited by the i and h values. We have opted for this solution to set the same fixed length of the chromosome, for all the individuals. Every time a new individual (genotype) is used to generate an ANN (phenotype) first the i and h parameters are read, in order to discard the unwanted extra columns and rows of the default matrix, leading to a matrix with the dimensions $(i'+1) \times h'$, where i' and h' denote the real number of input and hidden nodes. In most cases $i = i'$ and $h = h'$. Yet, in some rare cases it may occur that $i' < i$ or $h' < h$ if the last binary rows (or columns) are all set to zero.

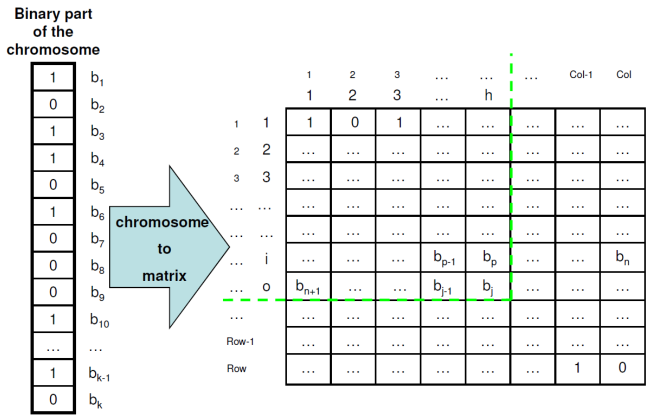


Figure 5: Example of the process to obtain a connection matrix from a given chromosome.

2.4. Time Lagged Selection Evolutionary Artificial Neural

This section presents the proposed Time lag selection EANN (TEANN), which is similar to the EANN of Section 2.2 accept that we now also search for the best set of time lags for the TLFN. As it was explained in Section 2.1, every time a new individual (i.e., ANN) is generated, training and validation patterns subset have to be obtained. In previous example (Fig. 1), if the ANN had k input nodes, all k previous values from the time series ($t-1$,$t-2$,...,$t-k$) were used to generate the patterns set. A new level of specialization is considered here, where the patterns are obtained by time lag filtering; i.e., selecting the relevant previous time lags of the series to generate the patterns to feed the ANN, which defines how the sliding window is set.

To carry out this new approach, more genes were added into the base chromosome. In particular, we adopted a binary codification, where each new gene defines if the time lag is (or not) used by the model. Fig. 6 shows the new codification scheme that includes time lag selection.

We set k to the maximum number of inputs (i.e., 100). Yet, it should be noted that i now sets the maximum number of input nodes, i.e., only

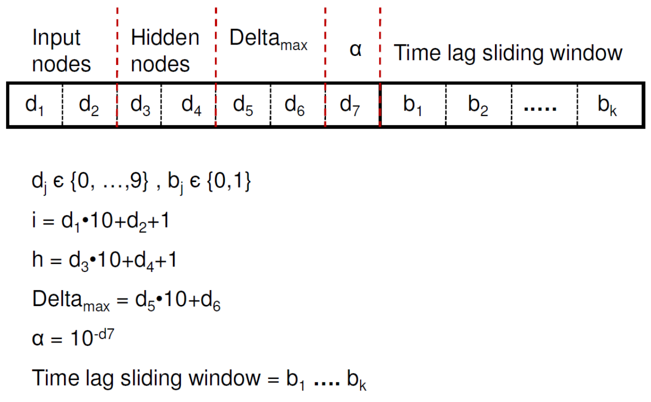

Input nodes
Hidden nodes
Delta$_{max}$
α
Time lag sliding window
d$_1$ d$_2$ d$_3$ d$_4$ d$_5$ d$_6$ d$_7$ b$_1$ b$_2$ ..... b$_k$
d$_j$ є {0, …,9} , b$_j$ є {0,1}
i = d$_1$•10+d$_2$+1
h = d$_3$•10+d$_4$+1
Delta$_{max}$ = d$_5$•10+d$_6$
α = 10^{-d7}
Time lag sliding window = b$_1$ .... b$_k$

Figure 6: Encoding scheme with time lag selection.

the up to b_i time lags are considered by the model. Lets considering the same example of Fig. 1, where $i = 3$, but now with the time lag selection where $b_1 = 1$, $b_2 = 0$ and $b_3 = 1$. As shown in Fig. 7, the number of input nodes of the ANN are set not only by $i = 3$, but also depends on the binary encoding, which only activates two lags, thus $I = 2$. Thus, the first pattern is $N_{t0}, N_{t2} \Rightarrow N_{t3}$.

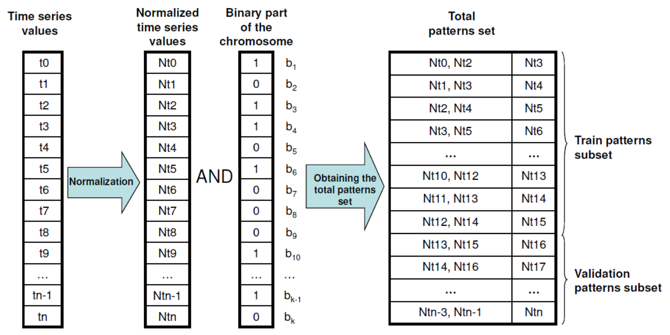

Time series values
Normalized time series values
Binary part of the chromosome
Total patterns set
Normalization
AND
Obtaining the total patterns set
Train patterns subset
Validation patterns subset

Figure 7: Process to obtain training and validation sets with time lag selection.

3. Experimental Setup and Results

3.1. Time Series

In this work, we selected a total of six time series, with different characteristics and from distinct domains. Five series were selected from the well-known Hyndman's time series data library repository [21]. These are named

Passengers, Temperature, Dow-Jones, Quebec and Abraham12. Passengers has the information about the number of passengers of an international airline in thousands, measured monthly from January of 1949 till December of 1960. Temperature is about the mean monthly of air temperature measured at Nottingham Castle from 1920 till 1939. Dow-Jones contains the monthly closings of the Dow-Jones industrial index from August of 1968 till August of 1981. Abraham12 represents gasoline demand at Ontario, in millions of gallons, from 1960 to 1975. Quebec includes the number of births, as daily measured in Quebec, from 1st of January of 1977 till 31 of December of 1978. We also adopt the Mackey-Glass series [16], which is a common benchmark for comparing the generalization ability of different forecasting methods. This series is a chaotic time series generated from a time-delay ordinary differential equation.

It should be noted that these six times series were also adopted by the *NN3* and *NN5* forecasting competitions [11]. Except for Mackey-Glass, all datasets are from real-world domains and such data can be affected by external issues (e.g., floods, strikes, technological advances), which make them interesting datasets and more difficult to predict.

3.2. Evaluation

The global performance of a forecasting model is evaluated by an accuracy measure, such as MSE, Relative Squared Error (RSE) and SMAPE:

$$\begin{aligned} MSE &= \tfrac{1}{H}\sum_{t=T+1}^{T+H} e_t^2 \\ RSE &= \frac{\sum_{i=T+1}^{T+H} e_i^2}{\sum_{i=T+1}^{T+H} (y_t-\overline{y_t})^2} \times 100\% \\ SMAPE &= \tfrac{1}{H}\sum_{t=T+1}^{T+H} \frac{|e_t|}{(|y_t|+|\hat{y}_t|)/2} \times 100\% \end{aligned} \tag{2}$$

where $e_t = y_t - \hat{y}_t$, T is the current time period and H is the forecasting horizon, the number of multi-step ahead forecasts. In all these measures, lower values indicate better forecasts. Historically, the MSE (as well as its root mean square error RMSE variant) is a very popular metric within TSF. RSE and SMAPE have the advantage of being scale independent, thus can be more easily used to compare methods across different series. A value of RSE lower than 100% shows that the forecasting model is better than the naive average predictor. Although the SMAPE was originally proposed in [4], Eq. 2 adopts the variant used in [3], since it does not lead to negative values, thus ranging from 0% to 200%. SMAPE was also the error metric used in *NN3*, *NN5* and *NNGC1* forecasting competitions [11]. In this paper, we compute both SMAPE and MSE metrics (RSE is used to average the squared errors through the different series). When comparing results, statistical significance is given in terms of the Diebold and Mariano test proposed in [14] under a $\alpha = 0.1$ level.

To evaluate the different forecasting methods, each series was initially divided into two sets: in-samples and out-of-samples. The in-sample data were used to build the forecasting models. In case of the EANN methods, such data is further divided into training and validation sets (Section 2.1). The out-of-samples, also known as test data, includes the more recent values and is used to evaluate the error metrics. Table 1 shows number of elements adopted for these sets.

Table 1: Time series in-sample/out-of-sample sizes

Time Series	**#in-samples**	**#out-of-samples** (H)
Passengers	125	19
Temperature	206	19
Dow-Jones	129	19
Abraham12	168	24
Quebec	735	56
Mackey-Glass	735	56

3.3. Additional Comparison Methods

For additional comparison purposes, we have chosen a conventional ARIMA approach (FP) and three recently proposed machine learning methods: Random Forest (RF), Echo State Network (ESN) and Support Vector Machine (SVM).

The ARIMA method is based on the implementation of the popular forecasting tool ForecastPro© (FP) [18], which is fed with the in-samples of the six datasets and the full automatic feature of the tool is executed to obtain the forecasts. The rationale is to use a popular benchmark that can easily be compared and that does not require expert model selection capabilities from the user.

The RF method was proposed by Breiman in 2001 [5] and it is based on an ensemble of T unpruned decision trees, where each tree is built by using a random feature selection with up to m features from bootstrap training samples. The predictions are built by averaging the outputs of T trees. RF is a nonlinear learning method that is more simpler to train and tune when compared with other ensemble approaches, such as boosting [19]. In the comparison, we use the default RF hyperparameters adopted by the randomForest package (e.g., $T = 500$, $m = I/3$) of the R tool [36], which tend to give good results in a wide range of problems. The default settings are more likely to be used by common (non expert) users, thus this seems a reasonable assumption for the comparison. The RF is set to use all time lags and the number of inputs is set to $I = K + 1$ [40], where K is the seasonal period of the time series. For the analyzed time series, we assume that K is known *a priori*. In this paper, and as followed in [40], we use autocorrelation values to find the cycle period, which is set to: $K = 12$ for Passengers, Temperature and Abraham 12; $K = 50$ for Dow-Jones; $K = 7$ for Quebec and $K = 30$ for Mackey-Glass.

The ESN is a new recurrent neural network type that was proposed by Maass et al. in 2002 [29]. ESN is capable of producing a fast nonlinear learning by using a large and random fixed recurrent network (called reservoir) and then combining the desired output signal by training a linear combination of all response signals. The main advantage of ESN for TSF is that only one input is required since it is a recurrent network, i.e., the inputs are fed according to the temporal sequence of the series and past patterns are stored in the memory of the network. To set the ESN, we adopt the same settings as the EANN methods, while the ESN hyperparameters are defined using the recommendations proposed in [28]. There is only one output (the predicted value) and multi-step ahead forecasts are built by iteratively using 1-ahead predictions as inputs (similarly to all EANN methods). The size of the reservoir was fixed to a large value of $N_x = 1000$. Also, both ESN hyperparameters, the learning rate α and regularization β, were set by using a two dimensional exponential grid search, within the range $\in \{2^{-10}, 2^{-9.5}, 2^{-9.0}, ..., 2^{0}\}$ (total of 21 searches per dimension). During the grid search, the same timely ordered holdout scheme used by the EANN methods was adopted, there the training data is split into 70% of the elements for training and the most recent 30% for validation. The validation set was used to select the α and β combination that resulted in the lowest MSE error. The ESN was implemented using the R tool code provided in [28].

SVM is a powerful learning tool that is based on a statistical learning theory and was developed in the 1990s due to the work of Vapnik and its collaborators [7]. It is based on two key concepts: using a kernel function SVM transforms input variables into a high dimensional feature space and then it finds the best hyperplane to model the data in the feature space. The SVM method is configured as described in [40]: it is based on a Gaussian kernel and the ϵ-insensitive loss function; time lags are selected using a sensitive analysis feature selection method and heuristics (e.g., grid search) are used to set the SVM hyperparameters. This method requires setting a maximum number of input time lags (I) for each series. For such purpose, the SVM adopts the same RF rule of $I = K + 1$, which was also used in [40]. The SVM was implemented in the R tool by using the rminer package [8].

3.4. Sparse ANN Results

Each EANN method explained in this paper was executed five times for each time series. In the experiments, we used a maximum of 100 generations as the stopping criterion of the EANNs. The forecasts were compared with the real ones and two error metrics were computed: MSE and SMAPE. The obtained results are shown in Table 2 (MSE errors) and Table 3 (SMAPE errors). In case of the EANN methods, we present the median of the five runs.

Table 2: SEANN forecasting comparison (MSE errors; best values in **bold**)

Series	**FP**	**RF**	**ESN**	**SVM**	**EANN**	**SEANN**
Pass. (MSE$\times 10^2$)	4.862	53.560	115.862	16.370	**4.207**	$4.381_{\text{FP,RF,ESN,SVM}}^{\star}$
Temp. (MSE)	4.383	4.922	7.073	4.929	**4.380**	$5.730_{\text{ESN}}^{\star}$
Dow-J. (MSE$\times 10^3$)	2.906	7.727	**2.640**	3.914	4.459	$3.435_{\text{RF,SVM,EANN}}^{\star}$
Abrah. (MSE$\times 10^8$)	2.512	3.321	10.352	2.149	3.023	$\mathbf{1.511}_{\text{FP,RF,ESN,SVM,EANN}}^{\star}$
Queb. (MSE$\times 10^2$)	9.304	15.494	12.165	11.494	12.579	$\mathbf{6.665}_{\text{FP,RF,ESN,SVM,EANN}}^{\star}$
Mack. (MSE$\times 10^{-3}$)	89.15	56.098	3.477	**0.218**	5.623	$1.593_{\text{FP,RF,ESN,EANN}}^{\star}$
RSE Average	61.4%	111.0%	108.1%	53.5%	58.7%	**38.7%**
RSE Median	63.4%	87.4%	97.4%	38.3%	36.1%	**20.4%**

* - statistically significant when compared with listed methods

An analysis to tables shows that the proposed SEANN provides in general better forecasts when compared with the baseline EANN. The sparsely connected method outperforms the fully connected one in four cases (with statistical significance), when considering both the MSE and SMAPE errors. A detailed look at SMAPE criterion, which is more important for the forecasting domain, shows interesting improvements of SEANN over EANN

Table 3: SEANN forecasting comparison (%SMAPE errors; best values in **bold**)

Series	**FP**	**RF**	**ESN**	**SVM**	**EANN**	**SEANN**
Passengers	4.51	11.80	18.27	6.00	3.39	**3.21**$_{\text{FP,RF,ESN,SVM}}^{\star}$
Temperature	**3.42**	**3.42**	4.61	3.66	3.51	4.17$_{\text{ESN}}^{\star}$
Dow-Jones	4.78	7.74	**4.77**	5.38	6.28	5.80$_{\text{RF,EANN}}^{\star}$
Abraham 12	6.21	7.00	12.21	5.53	6.42	**4.49**$_{\text{FP,RF,ESN,SVM,EANN}}^{\star}$
Quebec	10.37	12.92	11.50	11.56	10.83	**8.01**$_{\text{FP,RF,ESN,SVM,EANN}}^{\star}$
Mackey-Glass	30.40	18.61	6.40	**1.59**	7.07	4.03$_{\text{FP,RF,ESN,EANN}}^{\star}$
Average	9.95	10.25	9.63	5.62	6.25	**4.95**
Median	5.50	9.77	8.95	5.45	6.35	**4.33**

* - statistically significant when compared with listed methods

(e.g., a difference of 1.9, 2.8 and 3.0 pp for the last tree series). SEANN also outperforms other forecasting approaches, comparing favorable (with significance) against: FP in four cases for MSE and SMAPE; RF in five series for MSE and SMAPE; ESN in five cases for MSE and SMAPE; and SVM in four (MSE) and three series (SMAPE). The last two rows of the tables, which shows the average and median performance across all six time series, also ranks SEANN as the best method, under both error criteria. Table 4

Table 4: Comparison of the best models optimized by EANN and SEANN (smallest connection and time values in **bold**).

Series	**EANN**				**SEANN**				R_c	R_{te}
	inputs	hidden	connect.	time	inputs	hidden	connect.	time		
	(i)	(h)	(c)	(min)	(i')	(h')	(c)	(min)		
Passengers	49.2	67.4	3383.4	165	49.6	71.4	**1813.6**	**71**	46.4%	56.9%
Temperature	63.6	64.8	4186.1	315	65.0	59.8	**2011.1**	**114**	51.9%	63.8%
Dow-Jones	35.8	48.8	**1795.8**	161	38.6	93.6	1868.8	**73**	-4.1%	54.7%
Abraham12	30.4	117.8	3698.9	270	21.2	99.4	**1118.4**	**89**	69.8%	67.0%
Quebec	14.6	136.6	2131.0	6603	12.2	111.2	**824.8**	**5221**	61.3%	20.9%
Mackey-Glass	13.0	90.4	1265.6	8529	14.6	117.8	**924.6**	**5590**	26.9%	34.5%

compares the characteristics of the best ANNs evolved by both EANN approaches, where each cell shows the mean value of all five run executions. For each series and evolutionary method, we report the number of inputs (i or i'), hidden nodes (h or h'), total number of connections (c) and computational effort, given in terms of the total time elapsed for the EANN method (in *min*). The last two columns shows the reduction rate achieved when comparing SEANN against the base EANN, where $R_F = 1 - F_{\text{SEANN}}/F_{\text{EANN}}$ and F is the factor of analysis (i.e., c – total number of connections or te – time elapsed). The table shows that, in general, SEANN obtains simpler ANN structures. In particular, high reduction rates were achieved for Abraham12 and Quebec series. The exception occurs with the Dow-Jones dataset, where SEANN optimizes an ANN with much more hidden nodes when compared with EANN, although the final number of connections is only 4% higher than EANN. Furthermore, SEANN is always faster than EANN, requiring much less computation in all time series considered.

For demonstrative purposes, we analyze a detailed SEANN execution example for Passengers. It is assumed the user has only access to the in-samples, in this case 125 elements of the series. SEANN is executed and stopped after 100 generations. During SEANN execution, several ANNs are trained with the first 88 elements and fitness function is computed over the remaining 37 values. The fitness convergence process is shown in the left of Fig.8, in terms of MSE of best individual (when considering the normalized values, y-axis) and number of generations (x-axis). After one run, the best chromosome found was $(4, 9, 9, 2, 0, 5, 1, 0, 1, 0, 1, ...)$. This corresponds to an ANN with a maximum of $i = 50$ input nodes and $h = 93$ hidden nodes. The binary digits (e.g., 0,1,0,1) set the connectivity of the ANN as shown in Fig. 5 (e.g., the first input node connects to the second and fourth hidden nodes). In this case, a total of 2358 ANN connections were defined. This ANN is trained with the RPROP algorithm using the parameters $\Delta_{max} = 5$ and $\alpha = 1$ and all 125 Passengers elements. The last 50 values of the series are fed into the ANN, in order to obtain the first (1-ahead) forecast, which is 484 (close to the true series value of 472). The inputs are then slided, such that the oldest input is removed and 484 is fed as the last input, in order to obtain the 2-ahead forecast (536, also close to the true 548 value). The process is repeated until all 19 ahead forecasts have been executed, which are shown in Fig. 8 and that are close to the Passengers true values.

3.5. Time Lag Selection Results

Similarly to the previous section, each evolutionary approach (i.e., EANN and TEANN) was executed five times, with a stopping criterion of 100 generations, for all time series, and results are reported as the median of these five executions. The obtained results are shown in Tables 5 and 7.

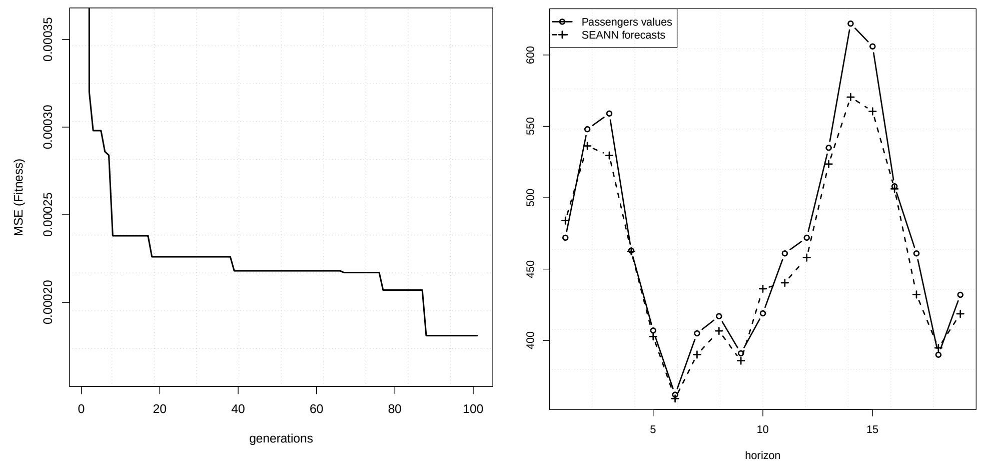


Figure 8: Example of SEANN convergence (left) and forecasts (right) for Passengers.

As it can be observed in Tables 5 and 7, the proposed time lag selection strategy (TEANN) outperforms with statistical significance the full time lag approach (EANN) in four cases for MSE and SMAPE. Also, the Passengers and Temperature error improvements of EANN over TEANN are rather small. For example, according to SMAPE, there is only a difference of 0.4 and 0.2 pp for Passengers and Temperature. In contrast, TEANN outperforms EANN by a pp difference that ranges from 0.7 to 2.1 for the remaining series. The average and median results, when considering all time series (last two rows of the tables), also favors TEANN when compared with EANN. When comparing of TEANN against other forecasting approaches, TEANN outperforms (with significance and for both MSE and SMAPE metrics): FP in 3 series; RF in 5 cases; ESN in 5 series; and SVM in 4 cases. Moreover, the average and median results (over all six series, last two rows of the tables), ranks TEANN as the best forecasting method for both error metrics.

As an example, we present the best individuals achieved by TEANN during a given execution in Table 8, where the third column (sliding window) shows the selected time lags (i.e., b_j values up to i). As shown by the fourth and fifth (number of input nodes used by the model) columns, the binary time lag selection genes perform a substantial pruning of the maximum number of input nodes (i), thus leading to much simpler models. The left of Fig. 9 shows the fitness convergence process (MSE of best individual when considering

Table 5: TEANN forecasting comparison (MSE errors; best values in **bold**).

Series	**FP**	**RF**	**ESN**	**SVM**	**EANN**	**TEANN**
Pass. (MSE$\times 10^2$)	4.862	53.560	115.862	16.370	**4.207**	5.476$_{\text{RF,ESN,SVM}}^{\star}$
Temp. (MSE)	4.383	4.922	7.073	4.929	**4.380**	5.501$_{\text{ESN}}^{\star}$
Dow-J. (MSE$\times 10^3$)	2.906	7.727	**2.640**	3.914	4.459	3.642$_{\text{RF,SVM,EANN}}^{\star}$
Abrah. (MSE$\times 10^8$)	2.512	3.321	10.352	2.149	3.023	**1.426**$_{\text{FP,RF,ESN,SVM,EANN}}^{\star}$
Queb. (MSE$\times 10^2$)	9.304	15.494	12.165	11.494	12.579	**8.450**$_{\text{FP,RF,ESN,SVM,EANN}}^{\star}$
Mack. (MSE$\times 10^{-3}$)	89.15	56.098	3.477	**0.218**	5.623	2.182$_{\text{FP,RF,ESN,EANN}}^{\star}$
RSE Average	61.4%	111.0%	108.1%	53.5%	58.7%	**42.4%**
RSE Median	63.4%	87.4%	97.4%	38.3%	36.1%	**20.3%**

* - statistically significant when compared with listed methods

Table 6: TEANN forecasting comparison (%SMAPE errors; best values in **bold**).

Series	**FP**	**RF**	**ESN**	**SVM**	**EANN**	**TEANN**
Passengers	4.51	11.80	18.27	6.00	3.39	**3.78**$_{\text{RF,ESN,SVM}}^{\star}$
Temperature	**3.42**	**3.42**	4.61	3.66	3.51	3.75$_{\text{ESN}}^{\star}$
Dow-Jones	4.78	7.74	**4.77**	5.38	6.28	5.54$_{\text{RF,SVM,EANN}}^{\star}$
Abraham 12	6.21	7.00	12.21	5.53	6.42	**4.34**$_{\text{FP,RF,ESN,SVM,EANN}}^{\star}$
Quebec	10.37	12.92	11.50	11.56	10.83	**9.30**$_{\text{FP,RF,ESN,SVM,EANN}}^{\star}$
Mackey-Glass	30.40	18.61	6.40	**1.59**	7.07	4.93$_{\text{FP,RF,ESN,EANN}}^{\star}$
Average	9.95	10.25	9.63	5.62	6.25	**5.27**
Median	5.50	9.77	8.95	5.45	6.35	**4.64**

* - statistically significant when compared with listed methods

the normalized values, y-axis) evolution through the number of generations (x-axis) for one execution of TEANN for the Passengers series. For this execution, the best evolved chromosome was $(4, 8, 8, 3, 5, 8, 6, 0, 0, 1, 1, 1, ...)$, which corresponds to an ANN with a maximum of $i = 49$, $h = 84$, trained with $\Delta_{max} = 58$ and $\alpha = 10^{-6}$. The binary digits set the time lags used. For instance, the set of {3,4,5} lags corresponds to the string $(0, 0, 1, 1, 1)$ and the full set of time lags is shown in the first row of Table 8 (total of 18 inputs). After trained with all 125 in-samples, the ANN was activated with the respective inputs, obtaining the values 491 and 536 for the first two ahead forecasts. The right of Fig. 9 plots the full 19 multi-step ahead forecasts.

Table 9 compares the characteristics of the best ANNs evolved by EANN and TEANN, where each cell shows the mean value of all five executions. For each series and evolutionary method, we report the number of inputs (I),

Table 7: TEANN forecasting comparison (%SMAPE errors; best values in **bold**).

Series	**FP**	**SVM**	**EANN**	**TEANN**
Passengers	4.51	6.00	3.39	**3.78**(SVM)*
Temperature	**3.42**	3.66	3.51	3.75
Dow-Jones	**4.78**	5.38	6.28	5.54(SVM,EANN)*
Abraham 12	6.21	5.53	6.42	**4.34**(FP,SVM,EANN)*
Quebec	10.37	11.56	10.83	**9.30**(FP,SVM,EANN)*
Mackey-Glass	30.4	**1.59**	7.07	4.93(FP,EANN)*
Average	9.95	5.62	6.25	**5.27**
Median	5.50	5.45	6.35	**4.64**

* - statistically significant when compared with the methods in parentheses

Table 8: Example of the best TEANN forecasting models.

Series	i	**sliding window**	**lag deletions**	**inputs**
Passengers	49	{3-5,9,13,15,16,18,22,... ,23,24,26,28,32,33,42,44,45,46,}	31	18
Temperature	67	{3-5,7,9-12,15,17,19,21,22,28,... ,34-36,38,41,46-49,53-57,59-61,65,66}	32	35
Dow-Jones	41	{3,4,7,8,11,13,17,22,... ,25-28,30,32,34,36,37,41}	23	18
Abraham12	30	{6,8,11-13,17-19,21,23,25,27,28}	17	13
Quebec	43	{2,5,8,10,14,16,17,23,... ,26,29,30,34,37,41-43}	26	17
Mackey-Glass	25	{6,8,10,11,13,15-17,19-25}	10	15

hidden nodes (h), total number of connections (c) and computational effort, given in terms of the total time elapsed for the EANN method (in *min*). The last two columns shows the reduction rate achieved when comparing TEANN against EANN, where $R_F = 1 - F_{\text{TEANN}}/F_{\text{EANN}}$ and F is the factor of analysis (i.e., c – total number of connections or te – time elapsed). In general, TEANN obtains simpler ANN structures. In particular, high reduction rates were achieved for Passengers and Abraham12 series. The exception is for Mackey-Glass, where TEANN optimizes an ANN with more hidden nodes when compared with EANN. Moreover, TEANN is always faster than EANN, requiring much less computation in all cases except Mackey-Glass.

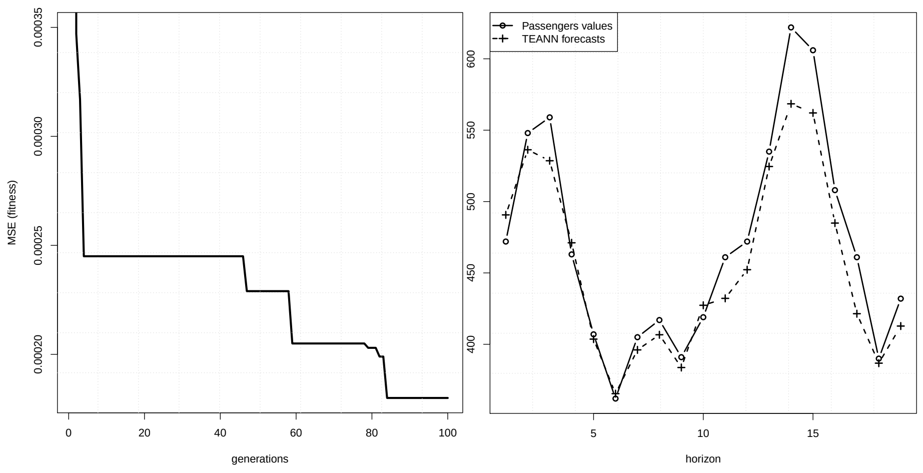


Figure 9: Example of TEANN convergence (left) and forecasts (right) for Passengers.

3.6. TEANN vs SEANN

In this section, we compare the results obtained by TEANN and SEANN. Each method tends to produce better results in around half of the series, as shown in Tables 10 (MSE values) and 11 (SMAPE). It should be noted the results of Table 10 and 11 need to be interpreted with caution, given that the differences are not statistically significant. Nevertheless, SEANN provides better global results (in terms of the best RSE average and SMAPE average and median overall values), with differences of 3.7 pp (RSE average) and 0.3 pp (SMAPE average and median).

On the other hand, Table 12 compares the characteristics of the best ANNs evolved by TEANN and SEANN, where each cell presents the mean value of all five executions. For each series and evolutionary method, we show the number of inputs (I), hidden nodes (h), total number of connections (c) and computational effort, given in terms of the total time elapsed (in min). The last two columns shows the reduction rate achieved when comparing SEANN against TEANN, where $R_F = 1 - F_{\text{SEANN}}/F_{\text{TEANN}}$ and F is the factor of analysis (i.e., c – total number of connections or te – time elapsed). The results of Table 12 show that for Passengers, Dow-Jones and Quebec series, TEANN requires less computational effort when compared with SEANN. For the first two of these series (Passengers and Dow-Jones), TEANN also optimizes ANN with much less inputs such that the total number of connections is also smaller when compared with the best model optimized by SEANN. For the remaining series, SEANN obtains a relevant improvement in terms

Table 9: Comparison of the best models optimized by EANN and TEANN (smallest connection and time values in **bold**).

Series	**EANN**				**TEANN**				R_c	R_{te}
	inputs	hidden	connect.	time	inputs	hidden	connect.	time		
	(I)	(h)	(c)	(min)	(I)	(h)	(c)	(min)		
Passengers	49.2	67.4	3383.4	165	23.0	72.0	**1728.0**	**61**	48.9%	63.0%
Temperature	63.6	64.8	4186.1	315	37.6	80.6	**3111.2**	**199**	25.7%	36.8%
Dow-Jones	35.8	48.8	1795.8	161	21.4	64.8	**1451.5**	**67**	19.2%	58.4%
Abraham12	30.4	117.8	3698.9	270	16.0	95.4	**1621.8**	**109**	56.2%	59.6%
Quebec	14.6	136.6	2131.0	6603	16.2	115.4	**1984.9**	**3906**	6.9%	40.8%
Mackey-Glass	13.0	90.4	**1265.6**	8529	12.0	120.4	1565.2	**8493**	-23.7%	0.4%

Table 10: SEANN vs TEANN comparison (MSE errors; best values in **bold**).

Series	**SEANN**	**TEANN**
Passengers (MSE$\times10^2$)	**4.381**	5.476
Temperature (MSE)	5.730	**5.501**
Dow-Jones (MSE$\times10^3$)	**3.435**	3.642
Abraham 12 (MSE$\times10^8$)	1.511	**1.426**
Quebec (MSE$\times10^2$)	**6.665**	8.450
Mackey-Glass (MSE$\times10^{-3}$)	**1.593**	2.182
RSE Average	**38.7%**	42.4%
RSE Median	20.4%	**20.3%**

of reduced number of connections and required computational effort, in particular for Temperature and Mackey-Glass. Overall, taking into account the average and median connection and time elapsed values (last two rows of Table 12), SEANN is faster than TEANN.

3.7. Computational Effort

This section analyzes the computational effort required by all forecasting approaches. The experimentation was carried out with an exclusive access to an Intel Xeon 2.27 GHz server. Table 13 shows the computational time (in minutes) required by each forecasting method. To facilitate the analysis, the methods were first ranked according to their average values (over all series) and then placed in the table from left (fastest method) to right (more

Table 11: SEANN vs TEANN comparison (%SMAPE errors; best values in **bold**).

Series	**SEANN**	**TEANN**
Passengers	**3.21**	3.78
Temperature	4.17	**3.75**
Dow-Jones	5.80	**5.54**
Abraham 12	4.49	**4.34**
Quebec	**8.01**	9.30
Mackey-Glass	**4.03**	4.93
Average	**4.95**	5.27
Median	**4.33**	4.64

Table 12: Comparison of the best models optimized by TEANN and SEANN (smallest connection and time values in **bold**).

Series	**TEANN**				**SEANN**				R_c	R_{te}
	inputs (i)	hidden (h)	connect. (c)	time (*min*)	inputs (i')	hidden (h')	connect. (c)	time (*min*)		
Passengers	23.0	72.0	**1728.0**	**61**	49.6	71.4	1813.6	71	-5.0%	-16.4%
Temperature	37.6	80.6	3111.2	199	65.0	59.8	**2011.1**	**114**	35.4%	42.7%
Dow-Jones	21.4	64.8	**1451.5**	**67**	38.6	93.6	1868.8	73	-28.7%	-9.0%
Abraham12	16.0	95.4	1621.8	109	21.2	99.4	**1118.4**	**89**	31.0%	18.3%
Quebec	16.2	115.4	1984.9	**3906**	12.2	111.2	**824.8**	5221	58.4%	-33.7%
Mackey-Glass	12.0	120.4	1565.2	8493	14.6	117.8	**924.6**	**5590**	40.9%	34.2%
Average			1910.4	2139			**1426.9**	**1860**		
Median			1674.9	154			**1466.0**	**102**		

computational demanding method).

The obtained results were expected, corresponding to the type of computational search performed by each method. The FP tool uses quick (but non disclosed) heuristics to find the best ARIMA method and requires the less computational effort. RF does not include any hyperparameter search and thus it is the second fastest method. SVM uses a linear search for finding the best set of time lags and it is the third fastest method. It should be noted that both RF and SVM require *a priori* knowledge for setting the cycle period value (K). Turning to the full automatic and generic methods, in forth place comes ESN, which uses a two dimensional grid search, followed

Table 13: Computational effort required by the forecasting methods (in *min*).

Series	**FP**	**RF**	**SVM**	**ESN**	**SEANN**	**TEANN**	**EANN**
Passengers	0.01	0.11	0.4	15	71	61	165
Temperature	0.01	0.02	0.1	17	114	199	315
Dow-Jones	0.01	0.01	0.6	15	73	67	161
Abraham 12	0.01	0.01	0.1	16	89	109	270
Quebec	0.02	0.11	0.1	31	5221	3906	6603
Mackey-Glass	0.02	0.11	0.5	31	5590	8493	8529
Average	0.01	0.06	0.3	21	1860	2139	2674
Median	0.01	0.07	0.3	17	102	154	293

by SEANN.

The proposed SEANN approach produces very competitive forecasts and it is faster than TEANN or EANN. While requiring more computational effort than non EANN based approaches, the time requirements are not limiting for small frequency time series (e.g., monthly). Moreover, the SEANN computational effort can be highly reduced by using parallel computation for each ANN training [17].

4. Conclusions

Time Series Forecasting (TSF) uses past patterns from a given event to forecast its future values. TSF is useful to support decision making in several domains (e.g., finance or production). In this paper, we approach multi-step ahead TSF using Artificial Neural Networks (ANNS). In particular, we adopt fully automatic Evolutionary ANN (EANN) search methods, which do not require prior knowledge from the user and that are based on evolutionary computation. As the evolutionary engine, we use the Estimation Distribution Algorithm (EDA), which has outperformed the standard genetic algorithm and differential evolution in our previous work [35]. Furthermore, we proposed two novel evolutionary design strategies: Sparsely connected Evolutionary ANN (SEANN) and Feature selection EANN (TEANN).

The two novel forecasting strategies (SEANN and TEANN) were compared over six distinct time series. As three baseline benchmarks, we also performed forecasts using the base EANN strategy, the popular ARIMA methodology and three recently proposed machine learning methods: Random Forest (RF), Echo State Network (ESN) and Support Vector Machine

(SVM). Also, the obtained multi-step forecasts were analyzed under two error criteria: MSE and SMAPE. The experiments held reveal the proposed SEANN and TEANN as the best forecasting methods. Moreover, when compared with the base EANN, both SEANN and TEANN tend to favor simpler structures and require less computational effort. Globally, when comparing SEANN and TEANN, similar predictive capabilities are achieved, although the former method tends to give better overall results, while requiring less computation effort and optimizing simpler strategies. Thus, we recommend SEANN as the best option for TSF with ANN.

In the future, we intend to explore a mixed TEANN-SEANN approach, by using the EANN to evolve both the time lags and the ANN connection weights. Such mixed approach is expected to optimize simpler ANN structures. In addition, the EDA search algorithm can be improved by using dependencies between its variables like Mutual Information Maximization for Input Clustering (MIMIC) [26] (i.e., variables with order one dependencies) or even *"tree"* EDA [26], with no restriction on the numbers of dependencies.

Acknowledgments

The research reported here has been supported by the Spanish Ministry of Science and Innovation under project TRA2010-21371-C03-03 and FEDER through the program COMPETE and the Portuguese Foundation for Science and Technology (FCT) within the project FCOMP-01-0124-FEDER-022674. The authors want to thank specially Martin Štěpnička and Lenka Vavříčková for all their help. The authors also want to thank Ramon Sagarna for introducing the subject of EDA.

References

[1] Cagdas Hakan Aladag, Ufuk Yolcu, Erol Egrioglu, and Ali Z Dalar. A new time invariant fuzzy time series forecasting method based on particle swarm optimization. *Applied Soft Computing*, 12(10):3291–3299, 2012.

[2] R.A. Aliev, B. Fazlollahi, RR Aliev, and B. Guirimov. Linguistic time series forecasting using fuzzy recurrent neural network. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 12(2):183–190, 2008.

[3] R.R. Andrawis and A.F. Atiya. A new Bayesian formulation for Holt's exponential smoothing. *Journal of Forecasting*, 28(3):218–234, 2009.

[4] J.S. Armstrong. *Long-range forecasting.* Wiley New York ETC., 1985.

[5] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] Y. Chen and Fi-John Chang. Evolutionary artificial neural networks for hydrological systems forecasting. *Journal of Hydrology*, 367(1-2):125 – 137, 2009.

[7] C. Cortes and V. Vapnik. Support Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[8] P. Cortez. Data Mining with Neural Networks and Support Vector Machines using the R/rminer Tool. In P. Perner, editor, *Advances in Data Mining – Applications and Theoretical Aspects, 10th Industrial Conference on Data Mining*, pages 572–583, Berlin, Germany, July 2010. LNAI 6171, Springer.

[9] P. Cortez, M. Rio, M. Rocha, and P. Sousa. Internet traffic forecasting using neural networks. In *Proceedings of The 2006 International Joint Conference on Neural Networks (IJCNN 2006)*, pages 4942–4949, Vancouver, Canada, IEEE Computer Society, jul 2002.

[10] P. Cortez, M. Rocha, and J. Neves. Evolving Time Series Forecasting ARMA Models. *Journal of Heuristics*, 10(4):415–429, 2004.

[11] Crone. Time Series Forecasting Competition for Neural Networks and Computational Intelligence. `http://www.neural-forecasting-competition.com`, acessed on January, 2011.

[12] Sven F Crone, Heiko Kausch, and D Prebmar. Prediction of the cats benchmark using a business forecasting approach to multilayer perceptron modelling. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 4, pages 2783–2788. IEEE, 2004.

[13] Sven F. Crone and Nikolaos Kourentzes. Feature selection for time series prediction - a combined filter and wrapper approach for neural networks. *Neurocomput.*, 73:1923–1936, June 2010.

[14] Francis X Diebold and Robert S Mariano. Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3):253–263, 1995.

[15] Francisco Fernández-Navarro, César Hervás-Martínez, Roberto Ruiz, and Jose C Riquelme. Evolutionary generalized radial basis function neural networks for improving prediction accuracy in gene classification using feature selection. *Applied Soft Computing*, 12(6):1787–1800, 2012.

[16] L. Glass and MC Mackey. Oscillation and chaos in physiological control systems. *Science*, 197:287–289, 1977.

[17] BPa Gonzalez, Juan Peralta Donate, Paulo Cortez, GGb Sánchez, and ASb De Miguel. Parallelization of an evolving artificial neural networks system to forecast time series using openmp and mpi. In *Evolving and Adaptive Intelligent Systems (EAIS), 2012 IEEE Conference on*, pages 186–191, Madrid, Spain, May 2012. IEEE.

[18] Robert L. Goodrich. The forecast pro methodology. *International Journal of Forecasting*, 16(4):533–535, 2000.

[19] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer-Verlag, NY, USA, 2nd edition, 2008.

[20] Simon Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, 1999.

[21] Hyndman. Time series data library. `http://robjhyndman.com/TSDL/`, acessed on January, 2011.

[22] Ashu Jain and Avadhnam Madhav Kumar. Hybrid neural network models for hydrologic time series forecasting. *Applied Soft Computing*, 7(2):585–592, 2007.

[23] N. Kasabov and Q. Song. Denfis: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Transactions on Fuzzy Systems*, 10:144–154, 2002.

[24] M. Khashei and M. Bijari. A novel hybridization of artificial neural networks and arima models for time series forecasting. *Applied Soft Computing*, 2010.

[25] A. Lapedes and R. Farber. Non-linear signal processing using neural networks: Prediction and system modelling. Technical report la-ur-87-2662, Los Alamos National Laboratory, USA, 1987.

[26] P. Larranaga and J. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation (Genetic Algorithms and Evolutionary Computation).* Springer, October 2001.

[27] S.T. Li, S.C. Kuo, Y.C. Cheng, and C.C. Chen. A vector forecasting model for fuzzy time series. *Applied Soft Computing*, 11(3):3125–3134, 2011.

[28] Mantas Lukoševičius. A practical guide to applying echo state networks. In *Neural Networks: Tricks of the Trade*, pages 659–686. Springer, 2012.

[29] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

[30] S. Makridakis, S.C. Wheelwright, and R.J. Hyndman. *Forecasting methods and applications.* John Wiley & Sons, USA, 3rd edition, 2008.

[31] R. Mendes, P. Cortez, M. Rocha, and J. Neves. Particle Swarms for Feedforward Neural Network Training. In *Proceedings of The 2002 International Joint Conference on Neural Networks (IJCNN 2002)*, pages 1895–1899, Honolulu, Havai, USA, IEEE Computer Society, May 2002.

[32] Geoffrey F. Miller, Peter M. Todd, and Shailesh U. Hegde. Designing neural networks using genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, pages 379–384, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[33] N. Moseley. Modeling Economic Time Series Using a Focused Time Lagged FeedForward Neural Network. In *Proceedings of Student Research Day, CSIS, Pace University*, May 2003.

[34] Harri Niska, Teri Hiltunen, Ari Karppinen, Juhani Ruuskanen, and Mikko Kolehmainen. Evolving the neural network model for forecasting air pollution time series. *Engineering Applications of Artificial Intelligence*, 17(2):159 – 167, 2004. Intelligent Control and Signal Processing.

[35] J. Peralta, G. Gutierrez, and A. Sanchis. Time series forecasting by evolving artificial neural networks using genetic algorithms and estimation of distribution algorithms. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1 –8, July 2010.

[36] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[37] M. Riedmiller. Supervised Learning in Multilayer Perceptrons - from Backpropagation to Adaptive Learning Techniques. *Computer Standards and Interfaces*, 16, 1994.

[38] M. Rocha, P. Cortez, and J. Neves. Evolution of Neural Networks for Classification and Regression. *Neurocomputing*, 70:2809–2816, 2007.

[39] J. David Schaffer, Richard A. Caruana, and Larry J. Eshelman. *Using genetic search to exploit the emergent behavior of neural networks*, pages 244–248. MIT Press, Cambridge, MA, USA, 1991.

[40] Martin Stepnicka, Paulo Cortez, Juan Peralta Donate, and Lenka Stepnicková. Forecasting seasonal time series with computational intelligence: On recent methods and the potential of their combinations. *Expert Systems with Applications*, 40(6):1981–1992, 2013.

[41] D Whitley, T Starkweather, and C Bogart. Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing*, 14(3):347 – 361, 1990.

[42] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[43] Andreas Zell, Günter Mamier, R. Hübner, N. Schmalzl, Tilman Sommer, and Michael Vogt. Snns: An efficient simulator for neural nets. In *MASCOTS '93: Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems*, pages 343–346, San Diego, CA, USA, 1993. Society for Computer Simulation International.

[44] Guoqiang Zhang, B. Eddy Patuwo, and Michael Y. Hu. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14(1):35–62, 1998.