Neural Comput & Applic (2013) 22:11–20
DOI 10.1007/s00521-011-0741-0

CONT. DEV. OF NEURAL COMPT. & APPLN.

Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm

Juan Peralta Donate · Xiaodong Li · Germán Gutiérrez Sánchez · Araceli Sanchis de Miguel

Received: 14 February 2011 / Accepted: 9 September 2011 / Published online: 14 October 2011

Abstract Time series forecasting is an important tool to support both individual and organizational decisions (e.g. planning production resources). In recent years, a large literature has evolved on the use of evolutionary artificial neural networks (EANN) in many forecasting applications. Evolving neural networks are particularly appealing because of their ability to model an unspecified nonlinear relationship between time series variables. In this work, two new approaches of a previous system, automatic design of artificial neural networks (ADANN) applied to forecast time series, are tackled. In ADANN, the automatic process to design artificial neural networks was carried out by a genetic algorithm (GA). This paper evaluates three methods to evolve neural networks architectures, one carried out with genetic algorithm, a second one carried out with differential evolution algorithm (DE) and the last one using estimation of distribution algorithms (EDA). A comparative study among these three methods with a set of referenced time series will be shown. In this paper, we also compare ADANN forecasting ability against a forecasting tool called Forecast Pro® (FP) software, using five benchmark time series. The object of this study is to try to improve the final forecasting getting an accurate system.

Keywords Evolutionary computation · Genetic algorithms · Differential evolution · Estimation of distribution algorithm · Artificial neural networks · Time series · Forecasting

J. P. Donate (✉) · G. G. Sánchez · A. S. de Miguel
Computer Science Department, Group CAOS, University Carlos III of Madrid, Avenida de la Universidad 30, 28911 Leganes, Spain
e-mail: jperalta@inf.uc3m.es

G. G. Sánchez
e-mail: ggutierr@inf.uc3m.es

A. S. de Miguel
e-mail: masm@inf.uc3m.es

X. Li
School of Computer Science and Information Technology, Royal Melbourne Institute of Techonology, Melbourne, VIC, Australia
e-mail: xiaodong.li@rmit.edu.au

1 Introduction

Forecasting the future based on the observed past is an important tool to support individual and organizational decision making. Time series forecasting (TSF), which predicts the behavior of a given phenomenon based solely on the past values of the same event, has become increasingly used in areas such as agriculture, finance, management, production or sales.

Several TSF methods have been proposed, such as Holt-Winters (in the sixties) or the ARIMA methodology [1] (in the seventies). More recently, several computational intelligence (CI) methods have been applied to TSF such as immune systems [2], support vector machines (SVM) [3], artificial neural networks (ANN) [4], fuzzy techniques [5] or hybrid systems combining any of the previous ones with evolutionary search [6, 7].

In this paper, we focus on ANN [11] which are flexible models that do not require a priori knowledge, are capable of nonlinear modeling and also often robust to noisy data. These properties of ANN make them a natural solution for TSF. In effect, ANN have been applied in real-world forecasting tasks, such as market predictions [8], meteorological [9] and network traffic forecasting [10].

A crucial issue is the design of an appropriate ANN model for a particular time series [4]. It will be necessary to

set the type of ANN that will solve the forecasting task (Multilayer Perceptron), the learning algorithm used (Backpropagation) and which specific architecture (topology and connection weights values) will be better to forecast this specific time series. So, it has to be established the suitable value for each degree of freedom in the ANN [11] (number of input nodes, number of outputs neurons, number of hidden layers, number of hidden neurons in each layer, the connections from one node to another, connection weights, etc.).

The goals of this paper are to develop an accurate automatic method, to design ANN and to forecast time series based on evolutionary computation (EC). This automatic process consists of an evolutionary search technique such as a GA that uses its exploitation and exploration properties to find a good solution. The chromosome codifies the set of parameters previously commented, and the fitness function will be minimum validation error obtained during the training process. On the other hand, two more EC methods will be used to carry out the evolutionary search, DE and EDA.

The paper is organized as follows. Section 2 reviews the related work about forecast task with ANN. Section 3 explains how our system designs ANN with GA, DE and EDA to forecast time series. In Sect. 4, experimental setup and results are shown. And finally, conclusions and future works are described in Sect. 5.

2 Related work

Several authors have addressed forecasting tasks using ANN [4, 7, 12, 13]. This reveals the full consideration of ANN (as a data driven learning machine) into forecasting theory [14, 15].

Often, the process of designing the correct ANN model for TSF is based on trial and error heuristics. If manual design of ANN is carried out, several topologies (i.e. different number of inputs nodes and different number of hidden neurons) with different learning rates are trained. For each of them, training and validation errors are obtained, and one with better generalization capability is selected to forecast the future values. A better alternative is to use an automatic ANN design, where EC plays an important role, in what is known as evolutionary ANN (EANN). Several works have proposed EANN, such as [16–20].

Some of them use direct encoding schemes (DES) [16, 17], and the others use Indirect Encoding Scheme (IES) [18–20]. For DES, the chromosome contains information about parameters of the topology, architecture, learning parameters, etc. of the artificial neural network. In IES the chromosome contains the necessary information, so that a constructive method gives rise to an artificial neural network topology (or architecture).

Abraham [21] shows an automatic framework for optimization ANN in an adaptive way, and Yao et al. [22] try to spell out the future trends of the field. About the use of EANN in time series forecasting, many works have been carried out [23–25]. In [14], Patuwo and Hu present a "state of the art" of ANN into forecasting task, and in [13], Crone proposes a stepwise selection of ANN models for time series forecasting.

Chen et al. [26] propose local linear wavelet neural network to forecast time series. Rivas et al. [27] use evolving RBF neural networks for time series forecasting. Few studies have been done using ANN and DE to generate a hybrid system to forecast time series [28, 29]. Since EDA is a more recent technique, its use for EANN in TSF is scarce and within our knowledge, has only appeared very recently. Therefore, the main motivation of this paper is to obtain ANN models through a fully automatic process, due to not all users who need to deal with forecasting are ANN experts. They are usually people from industry or economics, not computer scientists.

3 Evolutionary design of artificial neural networks

3.1 Time series and ANN

The problem of forecasting time series with ANN [7] is considered as obtaining the relationship from the value at period "t" (in this system, the resulting ANN will have only one output neuron) and the values from previous elements of the time series ($t-1$, $t-2$, …, $t-k$) to obtain a function as it is shown in (1):

$$a_t = f(a_{t-1}, a_{t-2}, \ldots, a_{t-k}) \tag{1}$$

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 of the time series are normalized into the range [0,1] (leading to the N_t values). Once the ANN gives 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 (i.e. 1 to N ahead forecasting) because if it was allowed several nodes at the output layer (t, $t+1$, …, $t+n$), the forecasting task would be done for several output future values in a row (1 ahead forecast). This forecasting method would lead to forecast value t from $t-k$, …, $t-2$, $t-1$ but also to forecast values $t+1, t+2, \ldots, t+n$ from $t-k, \ldots, t-2$, $t-1$, so in every forecasting step, important previous data would be missing for $t+1, t+2, \ldots, t+n$. Due to this, it is considered better to use 1 to N ahead forecast.

Therefore, the time series will be transformed into a pattern set depending on the k input nodes of a particular ANN, and each pattern will consist of the following:

- k inputs values that correspond to k normalized previous values of period t: N_{t-1},N_{t-2}, …, N_{t-k}.
- One output value: N_t (the desired target).

This patterns set will be used to train and validate each ANN generated during the evolutionary execution. Therefore, patterns set will be split into two subsets, training and validation. The complete patterns set is ordered into the same way the time series is. The first $x\%$ (where x is a parameter) from the total patterns set will generate the train patterns subset, and the validation subset will be obtained from the rest of the total patterns set. The test subset will be the future (and unknown) time series values that the user wants to forecast. An example of this process using an ANN with 3 input nodes ($k = 3$) can be seen at Fig. 1.

3.2 ADANN

The problem of designing ANN could be seen as a search problem into the space of all possible ANN. Then, that search can be done by a GA [30] using exploitation and exploration.

Therefore, there are three crucial issues: (1) about the solution's space, which information of the net should be included into the chromosome; (2) how this information is codified into the chromosome, i.e. encoding schema; and (3) how each individual is evaluated and translated into the fitness function.

In this approach, it has been chosen multilayer perceptron (MLP) as computational model due to its approximation capability and inside this group, Full Connected MLP with only a hidden layer and Backpropagation (BP) as learning algorithm, according to [31].

As it was mentioned in Sect. 1, the design of the ANN will be done by setting the parameters of the ANN. In the case of MLP with only one hidden layer and BP, these parameters are: number of inputs nodes, number of hidden neurons, number of output neurons (only one, it is set by the forecasting problem), which is the connection pattern (how the nodes are connected), and the whole set of connection weights (implemented by the seed used to initialize the connection weights as it will be explained below).

For our approach [7] to design ANN to forecast time series, a DES for Full Connected MLP has been considered. For this DES, the information placed into the chromosome will be two decimal digits, i.e. two genes, to codify the number of inputs nodes (i); other two for the number of hidden nodes (h); two more for the learning factor (α); and the last ten genes for the initialization seed value of the connection weights (s) (seed in SNNS [32] is "long int" type, that is why it has been used 10 genes (decimal digits) to encode "s"), as it can be seen in Fig. 2. This way, the values of "i," "h," "α" and "s" are obtained from the chromosome as it can be seen in (2). *Max_inputs* and *max_hidden* are parameters of the system. *NTS* represents the number of elements of the time series.

$$\begin{aligned} i &= NTS \cdot max\_inputs \cdot ((d_1 \cdot 10 + d_2)/100) \\ h &= NTS \cdot max\_hidden \cdot ((d_3 \cdot 10 + d_4)/100) \\ \alpha &= (d_5 \cdot 10 + d_6)/100 \\ s &= \sum_{j=7}^{16} d_j \cdot 10^{16-j} \end{aligned} \tag{2}$$

The search process (GA) will consist of the following steps:

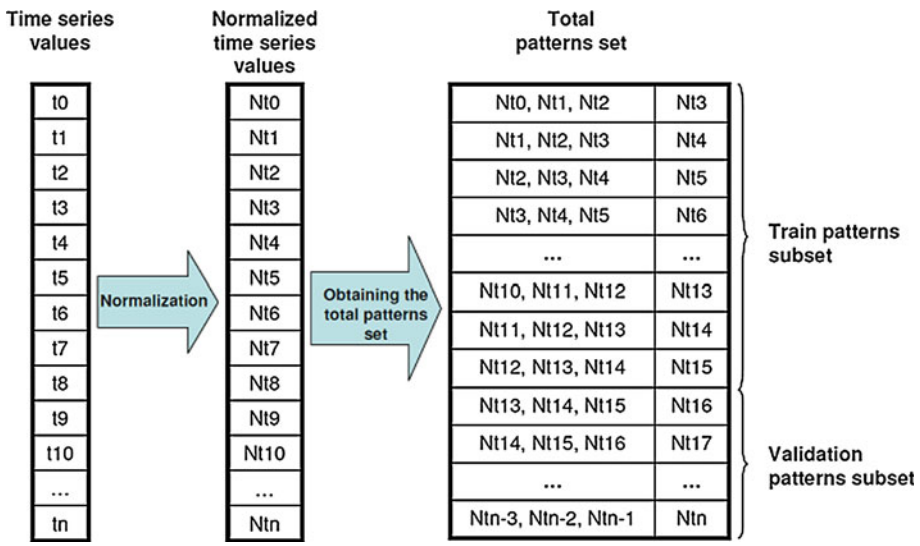

Fig. 1 Process to obtain train and validation pattern subsets

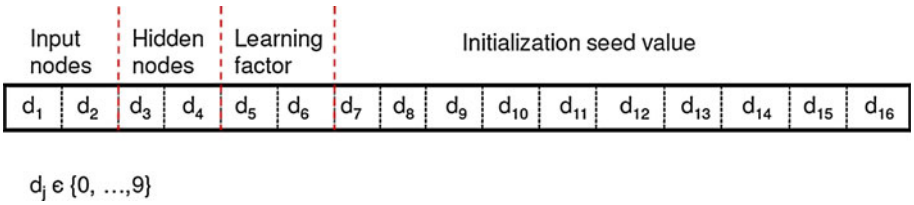

Fig. 2 Encoding practice used (decimal presentation) where d_j is the n decimal digit (0–9)

1. A randomly generated population, i.e. a set of randomly generated chromosomes, is obtained.
2. The phenotype (i.e. ANN architecture) and fitness value of each individual of the current generation are acquired. To obtain the phenotype associated to a chromosome and its fitness value:
   - 2.1. The phenotype of an individual of the actual generation is first obtained (using SNNS tool).
   - 2.2. 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 Sect. 3.1.
   - 2.3. The ANN is trained with BP (using SNNS). The architecture (topology and connections 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). The fitness for each individual is the minimum mean square error (MSE) during the learning process. So this architecture is the final phenotype of the individual.
3. Once that fitness values for whole population have been already obtained, the GA operators such as elitism, selection, crossover and mutation are applied in order to generate the population of the next generation, i.e. a new set of chromosomes.
4. Steps 2 and 3 are iteratively executed till a maximum number of generations are reached.

A schema of the whole search process can be seen at Fig. 3.

The fitness value for each individual is the minimum validation error during the learning process (ANN training), according to [43].

Regarding the use of MSE in the fitness function, the rationale is to reduce extreme errors that may highly affect multistep ahead forecasts. Also, preliminary experiments (using only training data and a few datasets) have shown that the MSE fitness leads to better forecasts when compared with mean absolute error (MAE). To choose the genetic algorithm parameters, Goldberg's GA [33] and [34] were taken into account, apart of the experience obtained during the preliminary experiments. The GA parameters were set to population size, *50*; maximum number of

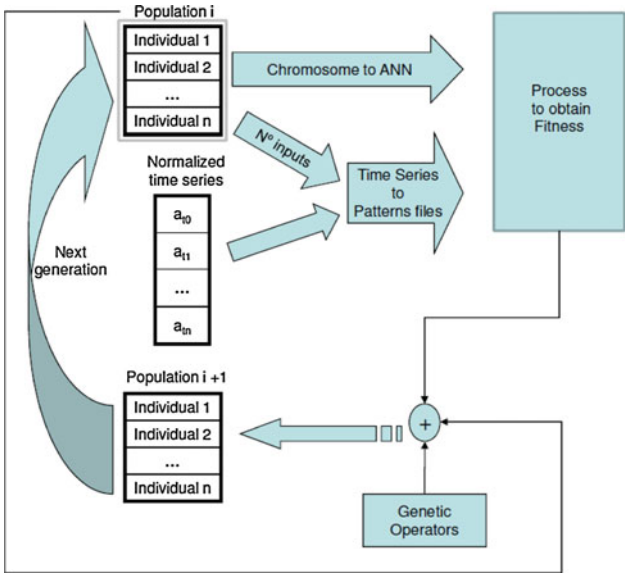

Fig. 3 ANN design by GA schema

generations, *100*; percentage of the best individual that stay unchangeable to the next generation (percentage of elitism), *10%*; crossover: parents are split in one point randomly selected, offspring are the mixed of each part from parents; mutation probability will be one divided by the length of the chromosome (*1/lengthchrom = 1/16 = 0.07*), and it will be carried out for each gene of the chromosome.

At the end of the search process, the best individual from the last generation is used to forecast the future unknown values $(t, t + 1, \ldots, t + n)$ one by one using the k previous known values $(t - k, \ldots, t - 2, t - 1)$. k is the number of inputs of the ANN obtained from the GA execution. To forecast several consecutive values $(t, t + 1, \ldots, t + n)$ every time a new value (t) is forecasted, it will be included in order into the previous known values set of the time series and used to forecast the next one, $t + 1$ (1 to N ahead forecast).

3.3 Differential evolution algorithm

Differential evolution algorithm (DE) is a stochastic nonlinear optimization algorithm by Storn and Price [35] and belongs to the class of evolution strategy optimizers. DE looks for the global minimum of a multidimensional, multimodal function trying to obtain a good probability.

DE community has been growing since the mid 1990s, and today more researchers are working on and with DE [29, 36].

The main idea of DE is a scheme for generating trial parameter vectors. DE differs from other evolutionary algorithms (EA) in their mechanism of generating offspring. In GA, an individual plays the role of a parent to generate an offspring. Nevertheless, DE adds the weighted difference between two vectors, i.e. chromosomes of the population, to a third vector. So, no separate probability distribution has to be used, which makes the scheme completely self-organizing.

In DE a population of solution vectors is successively updated by addition, subtraction and component swapping, until the population converges to an optimum. No derivatives are used, very few parameters are set and it is a simple and apparently very reliable method. That means DE does not require for the optimization problem to be differentiable as is required by classic optimization methods. DE can therefore also be used on optimization problems that are not even continuous and are noisy, change over time, etc.

Some studies using DE have been done in weather time series forecasting field [28, 29]. In this document, it is presented an automatic method to design ANN using advantages of DE (i.e. hybrid system), to forecast time series.

There are many schemes of generating individuals in DE. In this document, it is presented the most popular scheme, the *DE/rand/1/bin*, which is used in this research. Here, we have the process for a *DE/rand/1/bin*:

1. $P_{\text{initial}} \leq$ Generate and evaluate an initial population of x solutions.
2. Repeat for $i = 0, 1, 2, \ldots$, until a stopping criterion is met
   - 2.1. A target vector (X_i) and a base vector (X_{r0}) are chosen. The second one randomly, where i, r_0 ε $\{1, \ldots, x\}$.
   - 2.2. Two random different population members (X_{r1}, X_{r2}) are also chosen, where r_1, r_2 ε $\{1, \ldots, x\}$.
   - 2.3. Compute the difference vector from X_{r1} and X_{r2} (i.e. $X_{r1} - X_{r2}$).
   - 2.4. Multiply "*2.3*" by the mutation factor F (parameter of the system).
   - 2.5. $V_i \leq$ Add "*2.4*" to the base vector (X_{r0}) to obtain a mutant vector.
   - 2.6. $U_i \leq$ Crossover between target vector (X_i) and V_i.
   - 2.7. Selection is carried out between X_i and U_i.

To apply DE to our system, it was necessary to replace the GA, which is responsible for carrying out the global search into the hybrid system, by DE as it can be seen in [44].

3.4 Estimation distribution algorithm

Estimation of distribution algorithms (EDA) [42] is an outgrowth of GA. In GA, a population of individual solutions of a problem is kept as part of the search for a better solution. This population is typically represented as an array of objects. Depending on the specification of the GA, the objects might be bit strings, real numbers, etc. In EDA, this representation of the population is replaced by a probability distribution over the choices available at each position in the vector that represents each individual of the population.

For example, in the case the population is represented by bit vectors of length 4, the EDA for these populations would be a single vector of four probabilities (p1, p2, p3 and p4) where each p is the probability of that position being any of the possible values. Using this probability vector, it is possible to generate an arbitrary number of new candidate solutions.

In EC, new solutions are often generated by combining and modifying existing ones in a stochastic way. Probability distribution of new solutions over the space of all possible ones is usually not specified. In EDA, a probability distribution is used to approximate a population and new solutions are obtained by sampling this distribution. This has several advantages; one of them is to avoid premature convergence and being a more compact representation.

Maybe because estimation distribution algorithm is a recent technique (came into use just a few years ago), it has been carried out few hybrid studies (i.e. ANN + EDA) applied only to classification domains [37]. In this paper, it is proposed a new hybrid method using advantages of EDA to design ANN to forecast time series. Besides, it is a totally automatic method.

There are different kinds of EDA, but for our approach, it has been chosen UMDA, with no dependencies between variables, according to [37]. Here, we have the process for a general EDA:

1. $D_0 \leq$ Generate and evaluate an initial population of solutions
2. Repeat for $k = 0, 1, 2, \ldots$, until a stopping criterion is met
   - 2.1. $D_k^{\text{Sel}} \leq$ Select a subset of solutions from D_k
   - 2.2. $P_k(X) \leq$ Estimate the empirical probability distribution of D_k^{Sel}
   - 2.3. $D_k^{\text{New}} \leq$ Sample solutions from $P_k(X)$

2.4. $D_{k+1} \leq$ Create the new population with solutions from D_k^{New} and D_k^{Sel}
2.5. $D_k \leq D_{k+1}$
2.6. Evaluate solutions in D_k

To apply EDA to our approach, it was necessary to replace the GA (explained in Sect. 3.2), which is responsible for carrying out the global search into the hybrid system, by EDA. It can be observed that the principal differences between GA and EDA consist of steps "*2.2*" and "*2.3,*" where instead of carrying out crossover and mutation, it is estimated the empirical probability of each individual and sampled the solutions. A more detailed explanation of the system using EDA is given in [45].

4 Experimental setup and results

4.1 Time Series

The research presented in this paper was initially motivated by NN3 (2007) and NN5 (2008) time series competition [38]. To compare the proposed TSF methods, we selected 5 benchmark time series. Four of them are series from the well-known Hyndman's Time Series Data Library (TSDL) [46].

These time series are named Passengers, Temperature, Dow-Jones and Quebec. Passengers time series represents the number of passengers of an international airline in thousands, measured monthly from January 1949 till December 1960. Temperature time series shows the mean monthly of air temperature measured at Nottingham Castle from 1920 till 1939; in this case. Dow-Jones is about the monthly closings of the Dow-Jones industrial index from August 1968 till August 1981. Quebec represents the number of births daily measured in Quebec from January 1, 1977, till December 31, 1978. The last one called Mackey–Glass is based on the Mackey–Glass differential equation and is widely regarded as a benchmark for comparing the generalization ability of different methods. This series is a chaotic time series generated from a time-delay ordinary differential equation. This time series has been chosen in order to extend the experimental datasets by a different kind of a benchmark, i.e. by a time series that is not based on real-world data and that contains neither a trend nor a noise component.

4.2 Evaluation

The global performance of a forecasting model is evaluated by an accuracy measure, such as root mean squared error (RMSE) and symmetric mean absolute percentage error (SMAPE):

$$\text{RMSE} = \sqrt{\frac{1}{h}\sum_{t=T+1}^{T+h} e_t^2} \tag{3}$$

$$\text{SMAPE} = \frac{1}{h}\sum_{t=T+1}^{T+h} \frac{|e_t|}{(|y_t| + |F_t|)/2} \times 100\% \tag{4}$$

Where $e_t = y_t - F_t$ for $t = T + 1, \ldots, T + h$, "T" is the number of previous known elements, "h" denotes the forecasting horizon, y_t represents the real value and F_t the forecasted one.

In all these measures, lower values indicate better forecasts. Historically, the RMSE is a very popular metric within TSF.

SMAPE has the advantage of being scale independent, thus can be more easily used to compare methods across different series. Although the SMAPE was originally proposed in [39], (4) adopts the variant used in [40] since it does not lead to negative values (ranging from 0 to 200%). SMAPE is also used in NN3, NN5 and NNGC1 [38] forecasting competitions as evaluation error. For the forecasting comparison, we opted to compute SMAPE and RMSE.

4.3 Evolutionary method results

Each evolutionary approach (i.e. GA, DE and EDA hybrid systems) was executed five times, with a stopping criterion of 100 and 200 generations, for all time series. We report the mean results of these five executions. To evaluate the error for each method, forecasted values were compared with real values, under the RMSE and SMAPE criteria (3 and 4).

In Table 1, the results obtained for all time series and the average in generation number 100 are shown. On the other hand, it is shown the results obtained for all time series and the average in generation number 200 in Table 2.

As it can be observed in Table 1, applying DE instead of GA, we achieve better forecasting (SMAPE/RMSE) in two of the time series. On the problems of Mackey–Glass and Temperature, better SMAPE results are obtained with DE, and in Mackey–Glass case, the improvement is about *2.6%*. Besides, DE and EDA obtain better average results than GA.

If the experiments are run over 200 generations (Table 2), it can be seen an important improvement in almost all the time series, where DE obtains a better forecast than GA in four of the five time series. Only in Quebec time series, GA is still better than DE although difference is not significant. A special consideration has to be taken on Mackey–Glass time series where the SMAPE error result is *3.74%*, being the values forecasted by our approach almost identical to the real time series values.

Table 1 SMAPE and RMSE Passengers, Temperature, Dow-Jones, Quebec and Mackey–Glass with GA, DE and DEA for 100 generations (best values in bold)

100 Generations	GA	DE	EDA
Passengers			
SMAPE (%)	**3.18**	3.36	3.57
RMSE	$\mathbf{0.24 \times 10^{-1}}$	0.25×10^{-1}	0.26×10^{-1}
Temperature			
SMAPE (%)	4.31	**3.91**	3.98
RMSE	0.59×10^{-1}	$\mathbf{0.54 \times 10^{-1}}$	0.55×10^{-1}
Dow-Jones			
SMAPE (%)	**6.66**	8.19	6.81
RMSE	$\mathbf{0.14 \times 10^{0}}$	0.17×10^{0}	$\mathbf{0.14 \times 10^{0}}$
Quebec			
SMAPE (%)	**12.64**	13.74	13.27
RMSE	$\mathbf{0.15 \times 10^{0}}$	0.16×10^{0}	$\mathbf{0.15 \times 10^{0}}$
Mackey–Glass			
SMAPE (%)	8.67	**5.99**	6.27
RMSE	0.60×10^{-1}	$\mathbf{0.41 \times 10^{-1}}$	0.43×10^{-1}
Average			
SMAPE (%)	7.09	7.04	**6.78**
RMSE	0.86×10^{-1}	0.90×10^{-1}	$\mathbf{0.82 \times 10^{-1}}$

Table 2 SMAPE and RMSE Passengers, Temperature, Dow-Jones, Quebec and Mackey–Glass with GA, DE and DEA for 200 generations (best values in bold)

200 Generations	GA	DE	EDA
Passengers			
SMAPE (%)	3.15	**3.12**	3.22
RMSE	$\mathbf{0.23 \times 10^{-1}}$	$\mathbf{0.23 \times 10^{-1}}$	0.24×10^{-1}
Temperature			
SMAPE (%)	4.24	**3.91**	3.94
RMSE	0.58×10^{-1}	$\mathbf{0.54 \times 10^{-1}}$	0.56×10^{-1}
Dow-Jones			
SMAPE (%)	6.31	5.81	**5.26**
RMSE	0.13×10^{0}	0.12×10^{0}	$\mathbf{0.11 \times 10^{0}}$
Quebec			
SMAPE (%)	12.12	13.68	**10.96**
RMSE	0.14×10^{0}	0.16×10^{0}	$\mathbf{0.13 \times 10^{0}}$
Mackey–Glass			
SMAPE (%)	8.04	3.74	**1.81**
RMSE	0.55×10^{-1}	0.25×10^{-1}	$\mathbf{0.12 \times 10^{-1}}$
Average			
SMAPE (%)	6.77	6.05	**5.04**
RMSE	0.81×10^{-1}	0.76×10^{-1}	$\mathbf{0.66 \times 10^{-1}}$

Better results obtained by DE, compared with GA, after having run experiments 200 generations can be explained because in DE, more variation in population (because solutions still do not converge) leads to more varied search over solution space. That is why it can take more time to DE to arrive to a better solution.

As it can be seen in Table 1, EDA outperforms GA in two of the time series when the experiment has been run only 100 generations. Mackey–Glass and Temperature obtain better SMAPE results with EDA, and in Mackey–Glass case, the improvement is about *2.4%*. Besides, EDA does not win in any time series, but in the average, it is the best method.

But if the experiments are run over 200 generations, it can be seen in Table 2 an important improvement in almost all the time series, where EDA obtain a better forecast than GA in four of the five time series. Only in Passengers time series, GA is still better than EDA although both results are

really close. A special consideration has to be taken on Mackey–Glass time series where the SMAPE error result is *1.8%*, being the values forecasted by our method almost identical to the real time series values. On the other hand, EDA average with 200 generations is almost *1.7%* better than GA average. Good results obtained by EDA compared with GA after having run experiments 200 generations could be explained because EDA, unlike GA, avoid premature convergence. So, when GA gets stock in a local minimum value, EDA does not and it keeps on looking for a better result if it has more time, in this case, more generations to look for. In general, DE and EDA need more generations to improve results, but the improvements obtained after 200 generations, make these two methods a good option to be used as a global search in a hybrid system to design ANN to forecast time series.

To have a better idea how each method (i.e. GA, DE and EDA) has evolved during the 200 generations, it is shown in Fig. 4 the evolution of the fitness value for Mackey–Glass time series.

As it can be observed, a lower value means a better fitness. While GA stops decreasing its fitness value between generation 60 and 70, DE and EDA keeps on decreasing their values after generation 100.

4.4 ADANN versus Forecast pro®

As a baseline comparison, we choose the popular forecasting tool Forecast Pro® (FP) [41]. In particular, we fed the tool with the in-samples of five datasets (i.e. Passengers, Temperature, Dow-Jones, Quebec and Mackey–Glass) and executed the full automatic feature of the tool 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.

Table 3 shows the results obtained for each time series applying these two different methods, FP and ADANN using the EDA version with 200 generations. Error obtained by each method will be measured in SMAPE and RMSE as it has been done before.

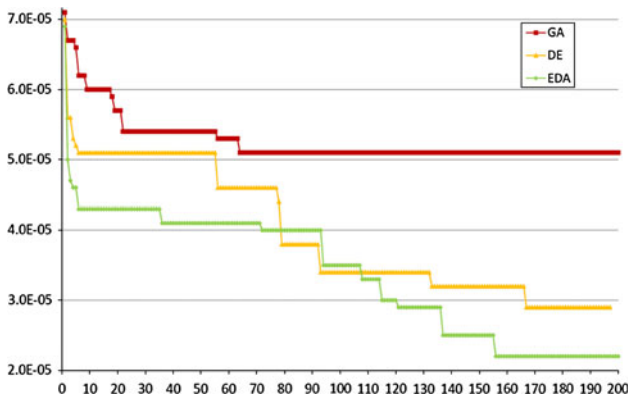

Fig. 4 GA, DE and EDA fitness value during 200 generations for Mackey–Glass time series

Table 3 SMAPE and RMSE Passengers, Temperature, Dow-Jones, Quebec and Mackey–Glass with ADANN and Forecast Pro®

	ADANN	*Forecast Pro®*
Passengers		
SMAPE (%)	**3.22**	4.51
RMSE	$\mathbf{0.24 \times 10^{-1}}$	0.27×10^{-1}
Temperature		
SMAPE (%)	3.94	**3.42**
RMSE	0.56×10^{-1}	$\mathbf{0.49 \times 10^{-1}}$
Dow-Jones		
SMAPE (%)	5.26	**4.78**
RMSE	$\mathbf{0.11 \times 10^{0}}$	$\mathbf{0.11 \times 10^{0}}$
Quebec		
SMAPE (%)	10.96	**10.37**
RMSE	0.13×10^{0}	$\mathbf{0.12 \times 10^{0}}$
Mackey–Glass		
SMAPE (%)	**1.81**	26.2
RMSE	$\mathbf{0.12 \times 10^{-1}}$	0.19×10^{0}
Average		
SMAPE (%)	**5.04**	9.85
RMSE	$\mathbf{0.66 \times 10^{-1}}$	0.99×10^{-1}

Different information can be acquired from Tables 1 and 2. First of all, Mackey–Glass forecast done by FP is not a good one, while ADANN obtains a SMAPE error of *1.8%*, really close to the real values. Although FP outperforms ADANN in three of the five time series (Temperature, Dow-Jones and Quebec), results from both systems are really close. Besides, ADANN has a better average.

5 Conclusions and future works

In this paper, we propose three methods to design ANN to forecast time series.

As they are totally automatic methods, any previous knowledge from the user is not required. The user does not have to be an expert in time series. The user just have to give the time series he wants to forecast and the number of future elements he wants to be forecasted to the system, and this method will give these forecasted values as result to the user.

The results of the experiments disclose that using DE and EDA, instead of GA, obtain different results depending on the number of generations they are executed. With only 100 generations, DE and EDA results do not improve too much compared to GA although both averages are better than GA. But if 200 generations are reached, it can be

observed a significant improvement, sometimes with a gain of *4.3%* in the results, as it happens in Mackey–Glass time series using DE and a gain of *6.2%* using EDA.

As it was commented before, obtaining better results by DE and EDA than with GA after having run the experiments 200 generations could be explained because in DE more variation in population (because solution has not converged yet) leads to more varied search over solution space. That is why it can take more time to DE to arrive to a better solution. When GA gets stuck in a local minimum, EDA does not and it keeps on looking for a better result if it is given more time, in this case, more generations to look for.

In general, DE and EDA need more generations to improve results better than GA, but the improvement obtained after 200 generations, make these two methods better options to use than GA in the global search for a hybrid system to design ANN to forecast time series.

ADANN, using its EDA and 200 generations version, outperforms FP forecasting tool in two of the five cases and obtains a better average.

Future works with additional time series, with similar characteristics to Quebec and Mackey–Glass, will allow us to obtain more accurate conclusions about the effect of using DE and EDA instead of GA. On the other hand, it would be really interesting to improve DE system with some ideas such as instead of using a random X_{r0}, use the best one (i.e. the one with the best fitness value); or instead of using single difference (i.e. $X_{r1} - X_{r2}$), use more vectors for more variations, for example ($X_{r1} - X_{r2} + X_{r3} - X_{r4}$). Or to improve EDA system with some ideas such as using EDA with dependencies between its variables such as MIMIC (i.e. variables with order one dependencies) or even "*tree*" EDA, 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 TRA2007-67374-C02-02. The authors want to thank specially Ramon Sagarna for introducing the subject of EDA.

References

1. Makridakis S, Wheelwright S, Hyndman R (2008) Forecasting methods and applications, 2nd edn. Wiley, USA
2. Nunn I, White T (2005) The application of antigenic search techniques to time series forecasting. GECCO. In: Preparation of papers for international journal of automation and computing 13, pp 353–360
3. Cortez P (2010) Sensitivity analysis for time lag selection to forecast seasonal time series using neural networks and support vector machines. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2010), pp 3694–3701, Barcelona, Spain
4. Crone S, Kourentzes N (2010) Feature selection for time series prediction a combined filter and wrapper approach for neural networks. Neurocomputing 73:1923–1936
5. Štěpnička M, Dvořák A, Pavliska V, Vavříčková L (2011) A linguistic approach to time series modeling with help of the F-transform. Fuzzy Sets Syst 180(1):164–184. doi:10.1016/j.fss.2011.02.017
6. Kasabov N, Song Q (2002) Dens: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE Trans Fuzzy Syst 10:144–154
7. Peralta J, Gutierrez G, Sanchis A (2008) Adann: automatic design of artificial neural networks. In Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation, GECCO'08, pp 1863–1870, New York, NY, USA
8. Hassan R, Nath B, Kirley M (2007) A fusion model of HMM, ANN and GA for stock market forecasting. Expert Systems with Applications 33:171–180
9. Maqsood I, Khan MR, Abraham A (2007) An ensemble of neural networks for weather forecasting. Neural Comput Appl 13(2):112–122
10. Bengio S, Fessant F, Collobert D (1995) A connectionist system for medium term horizon time series prediction, In: Proc int workshop application neural networks to telecoms, pp 308–315
11. Simon H (1998) Neural networks: a comprehensive foundation, 2nd edn. Prentice Hall, New Jersey
12. Ferreira TA, Vasconcelos GC, Adeodato PJ (2008) A new intelligent system methodology for time series forecasting with artificial neural networks. Neural Process Lett 28(2):113–129. doi:10.1007/s11063-008-9085-x
13. Crone SF (2005) Stepwise selection of artificial neural networks models for time series prediction. Journal of Intelligent Systems, Department of Management Science Lancaster University Management School Lancaster, United Kingdom
14. Zhang G, Patuwo BE, Hu MY (1998) Forecasting with artificial neural networks: the state of the art. Int J Forecast 14:35–62
15. Cortez P, Rocha M, Neves J (2004) Evolving time series forecasting ARMA models. J Heuristics 10(4):415–429. doi:10.1023/B:HEUR.0000034714.09838.1e
16. T. Ash. Dynamic Node Creation in Backpropagation Networks ICS Report 8901, The Institute for Cognitive Science, University of California, San Diego (Saiensu-sh, 1988), 1988
17. Fogel DB, Fogel LJ, Porto VW (1990) Evolving neural network. Biol Cybern 63:487–493
18. Gruau F (1992) Genetic synthesis of Boolean neural networks with a cell rewriting developmental process. Proceedings of COGANN-92 International Workshop on Combinations of Genetic Algorithms and Neural Networks, pp 55–74, IEEE Computer Society Press, Los Alamitos
19. Yao X, Lin Y (1997) A new evolutionary system for evolving artificial neural networks. Transactions on Neural Networks 8(3):694–713
20. Kitano H (1990) Designing neural networks using genetic algorithms with graph generation system. Complex Systems 4:461–476
21. Abraham A (2004) Meta-learning evolutionary artificial neural networks. Neurocomputing Journal 56c:1–38
22. Yao X (1993) A review of evolutionary artificial neural networks. International Journal of Intelligent Systems 8(4):539–567
23. Cortez P, Rocha M, Neves J (2006) Time series forecasting by evolutionary neural networks. In: Rubuñal J, Dorado J (eds) Artificial Neural Networks in Real-Life Applications. chapter III. Idea Group Publishing, Hershey, pp 47–70
24. Niska H, Hiltunena T, Karppinenb A, Ruuskanena J Kolehmainena M (2004) Evolving the neural network model for forecasting air pollution time series. Eng Appl Artif Intell 17(2):159–167. ISSN: 0952-1976
25. Chena Y-H, Chang F-J (2009) Evolutionary artificial neural networks for hydrological systems forecasting. J Hydrol 367(1–2):125–137. ISSN: 0022-1694

26. Chen Y, Yang B, Dong J (2005) Time-series prediction using a local linear wavelet neural network. Neurocomputing 69(4–6): 449–465
27. Rivas VM, Merelo JJ, Castillo PA, Arenas MG, Castellano, JG (2003) Evolving RBF neural networks for time-series forecasting with EvRBF. Inform Sci 165(3–4):207–220
28. Araújo RA, Vasconselos GC, Ferreria AE (2007) Hybrid differential evolutionary system for financial time series forecasting. In: IEEE congress on evolutionary computation, pp 4329–4336
29. Abdul-Kader HM (2009) Neural networks training based on differential evolution algorithm compared with other architectures for weather forecasting 34. Int J Comput Sci Netw Secur 9(3):92–99
30. Fogel D (1998) Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. Wiley-IEEE Press, New Jersey
31. Cybenko G (1989) Approximation by superposition of a sigmoidal function. Math Control Signals Syst 2:303–314
32. Prof. Dr. Andreas Zell,, WSI Computer Science Department, Computer Arquitecture, Software, Artif Neural Netw http://www-ra.informatik.uni-tuebingen.de/SNNS/
33. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Professional, Boston. ISBN 978-0201157675
34. http://www.obitko.com/tutorials/genetic-algorithms/parameters.php. November 2010
35. Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Global Opt 11:341–359
36. Wickramasinghe W, Li X (2008) Choosing leaders for multiobjective PSO algorithms using differential evolution, in Proceedings of the seventh international conference on simulated evolution and learning (SEAL'08), Lecture Notes in Computer Science (LNCS 5361), Springer, Berlin, pp 249–258
37. Cotta C, Alba E, Sagarna R, Larrañaga P (2002) Adjusting weights in artificial neural networks using evolutionary algorithms. In: Larrañaga P, Lozano JA (eds) Estimation of distribution algorithms. A new tool for evolutionary computation. Kluwer, Dordrecht, pp 357–373
38. Time Series Forecasting Competition for Neural Networks and Computational Intelligence. http://www.neural-forecasting competition.com. Accessed on October 2010
39. Armstrong J (1985) Long-range forecasting. Wiley, New York
40. Andrawis R, Atiya A (2009) A new Bayesian formulation for Holt's exponential smoothing. Journal of Forecasting 28(3): 218–234
41. http://www.forecastpro.com/products/overview/index.htm. Accessed on December 2010
42. Larrañaga P, Lozano JA (2001) Estimation of Distribution Algorithms. A new tool for evolutionary computation (genetic algorithms and evolutionary computation). Springer, Berlin
43. David Schaffer J, Caruana RA, Eshelman LJ (1991) Using genetic search to exploit the emergent behavior of neural networks. MIT Press, Cambridge, pp 244–248
44. Peralta J, Li X, Gutierrez G, Sanchis A (2010) Time series forecasting by evolving artificial neural networks using genetic algorithms and differential evolution. In Proceedings of the 2010 WCCI conference, IJCNN-WCCI'10, pages 3999–4006, Barcelona, Spain
45. Peralta J, Gutierrez G, Sanchis A (2010) Time series forecasting by evolving artificial neural networks using genetic algorithms and estimation of distribution algorithms. In Proceedings of the 2010 WCCI conference, IJCNN-WCCI'10, Barcelona, Spain
46. Hyndman RJ Time series data library, http://robjhyndman.com/TSDL/. Accessed on June 2011