Time series forecasting using a weighted cross-validation evolutionary artificial neural network ensemble

Juan Peralta Donate[a,*], Paulo Cortez[b], Germán Gutiérrez Sánchez[a], Araceli Sanchis de Miguel[a]

[a] *Computer Science Department, Carlos III University, Avda. de la Universidad, 30. 28911 Legans, Madrid, Spain*
[b] *Centro Algoritmi, Departamento de Sistemas de Informação, Universidade do Minho, Campus de Azurém, 4800-058 Guimarães, Portugal*

Abstract

The ability to forecast the future based on past data is a key tool to support individual and organizational decision making. In particular, the goal of Time Series Forecasting (TSF) is to predict the behavior of complex systems by looking only at past patterns of the same phenomenon. In recent years, several works in the literature have adopted Evolutionary Artificial Neural Networks (EANN) for TSF. In this work, we propose a novel EANN approach, where a weighted n-fold validation fitness scheme is used to build an ensemble of neural networks, under four different combination methods: mean, median, softmax and rank-based. Several experiments were held, using six real-world time series with different characteristics and from distinct domains. Overall, the proposed approach achieved competitive results when compared with a non-weighted n-fold EANN ensemble, the simpler 0-fold EANN and also the popular Holt-Winters statistical method.

Keywords: Ensembles, Evolutionary Computation, Genetic Algorithms, Multilayer Perceptron, Time Series Forecasting.

*Corresponding author. Tel.: +34 91 624 9424; Fax: +34 91 624 9129.
Email addresses: jperalta@inf.uc3m.es (Juan Peralta Donate), pcortez@dsi.uminho.pt (Paulo Cortez), ggutierr@inf.uc3m.es (Germán Gutiérrez Sánchez), masm@inf.uc3m.es (Araceli Sanchis de Miguel)

1. Introduction

The use of Soft Computing Models in Industrial and Environmental Applications (SCMIEA) is a key research field due to its effectiveness in human life [1, 2, 3, 4, 5]. In particular, Time Series Forecasting (TSF), the prediction of a time ordered variable, is an important SCMIEA domain. In effect, TSF is often used to support decision making (e.g. planning production resources) in distinct industrial and environmental areas, such as agriculture, finance, management, sales and control [6]. Due to its importance, several TSF methods have been developed, mainly statistical methods (e.g. Holt-Winters) [7]. However, these classical statistical methods were developed decades ago, when there were higher computational restrictions (e.g. memory and computational power). More recently, Soft Computing based methods, such as fuzzy techniques [8] and Artificial Neural Networks (ANN) [9, 10, 11] were proposed for TSF. In this paper, we focus on ANNs, which are natural candidates since they are flexible nonlinear models and they do not require the use of a priori knowledge. Since time series often exhibit noise and nonlinear components (e.g. due physical processes), ANNs have the potential to outperform classic TSF methods, as shown by several works [9, 10, 11].

Modeling ANN for TSF involves the design of the network structure (e.g. number of input and hidden nodes) and setting of the training algorithm parameters. If a manual design is carried out, several ANN setups (e.g. with different number of inputs neurons and learning rates) need to be tested. Typically, this involves a model selection phase, where several ANNs are trained in order to select the best configuration (i.e. with higher generalization capacity) to forecast the future values. As an alternative, automatic design methods have been proposed. In particular, EANN algorithms are becoming a popular solution, since they perform a global multipoint search, quickly locating areas of high quality, even when the search space is very complex [12, 13, 14]. Given the interest in TSF, EANNs have been rapidly applied to this domain [10]. As EANN engine, in this work, we adopt the recently proposed Automatic Design of ANN (ADANN) system [11], which evolves a single ANN (0-fold) and obtains good forecasting results.

Another crucial issue when modeling ANNs for TSF, is related with the quality of the data provided to the ANN. This issue is particularly relevant when dealing with short time series (i.e with few elements), as there are less training patterns and it is more difficult for the ANN to learn how to gen-

eralize well. In this paper, we focus on this second issue, by proposing the use of a fitness weighted n-fold cross-validation learning scheme to create an ensemble of ANNs. Such scheme has the advantage of allowing the forecasting system to obtain more training patterns and thus may be more suited for producing more accurate forecasts when modeling short time series. To combine the individual ANN responses, we test four combination methods (mean, median, softmax and rank-based combinations). Furthermore, we compare the proposed approach with non-weighted n-fold EANN ensembles, the simpler 0-fold EANN and a classic statistical method, the Holt-Winters method.

The paper is organized as follows. Section 2 describes how to address TSF tasks with EANN and weighted cross-validation ensembles. Next, we describe the experiments held and analyze the obtained results (Section 3). Finally, closing conclusions are drawn in Section 4.

2. Time series forecasting with artificial neural networks

2.1. Automatic design of artificial neural networks

We adopt the fully connected multilayer perceptron with logistic activation functions. Let $y_1, y_2, \ldots, y_t$ denote the time series to be modeled. When using a feedforward ANN, TSF is achieved by using a sliding time window, according to [9, 10]:

$$y_t = ANN(y_{t-1}, y_{t-2}, \ldots, y_{t-k}) + e_t \tag{1}$$

where $\{t\text{-}1, t\text{-}2, \ldots, t\text{-}k\}$ is a set of time lags used, t is the current time period, ANN is the function modeled by the ANN, $\hat{y}_t$ is the value predicted by the ANN and $e_t = y_t - \hat{y}_t$ is the forecasting error. As the first step, the original values of the time series need to be normalized (within [0,1]). After training, the inverse process is carried out, transforming the ANN responses back to the original scale. Only one neuron is chosen at the output layer. Multi-step forecasts (1 to N ahead forecasts) are built by iteratively using 1-ahead predictions as inputs. By adopting a sliding time window of size k, the time series is transformed into a pattern set, where each pattern consists of:

- k input values - corresponding to the k normalized previous values: y_{t-1}, y_{t-2}, $\ldots$, y_{t-k}.
- one output value - corresponding to the normalized time series at period t that is forecasted.

To access the generalization ANN capability, the pattern set is split into training and validation data. The former, with first $x\%$ of the pattern set elements (training), is used to adjust the connection weights of the ANN, while the latter (validation), with the remaining patterns, is used to estimate the ANN generalization capabilities.

The problem of designing the best ANN can be seen as a search problem within the space of all possible setups. EANN systems use evolutionary computation algorithms to perform this search, such as Genetic Algorithms (GA) [15], which use both exploitation and exploration. Recently, we have proposed an EANN system for TSF, called ADANN [16] and that works as follows:

1. Each chromosome consists of 16 decimal digits (from 0 to 9), where: the first two digits set the number inputs of the ANN, the next two digits set the number of hidden nodes, the next two digits set up the backpropagation learning rate ($\eta = (d_1 \cdot 10 + d_2)/100$, $d_i \in \{0, ..9\}$); finally, the remaining digits set the random initialization seed of the ANN (useful to avoid local minima and store the best ANN).

2. A randomly generated population (with 50 individuals) is obtained.

3. The phenotype, or ANN architecture, and fitness value of each individual of the current generation is obtained. To obtain the phenotype associated to a chromosome and its fitness value:

   (a) The topology of an individual i from the actual generation is obtained.

   (b) Then, the training and validation patterns subsets for this ANN i are obtained from time series data.

   (c) Once each ANN is initialized with its topology and connection weights values with the information from the chromosome, it is trained with the backpropagation algorithm, using an early stopping scheme that stores the best ANN [17] and with a maximum of 5000 epochs. The validation pattern subset is used to estimate the ANN generalization capability, obtaining its fitness value. In this paper, this fitness is given by the Mean Squared Error (MSE). The aim is to reduce extreme errors (e.g. outliers) that may highly affect multi-step ahead forecasts.

4. Once the fitness values for whole population have been already obtained, the GA operators such as elitism, selection, one-point crossover and mutation (with a *0.07%* rate) are applied in order to generate the population of the next generation, translated into a new set of chromosomes.

The steps 3 and 4 are iteratively executed until a maximum number of generations is reached. Finally, the best individual from the last generation is used to compute the 1 to N ahead forecasts.

2.2. Cross-validation ensembles

Cross-validation, sometimes called rotation estimation, is a technique for assessing how a learning model will generalize to an independent data set. It is mainly used when the goal is prediction and one wants to estimate how accurately a predictive model will perform in practice. One round of cross-validation involves partitioning a sample of data into complementary subsets, fitting the model on one subset (called training set) and validating predictive results on the other subset (called validation set). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are aggregated over all rounds. Cross-validation is important in guarding against testing hypotheses suggested by the data (called Type III errors [18]), especially when further samples are hazardous, costly or difficult to collect. Cross-validation has been used in previous TSF works [19, 20]. In [19] it was used to determine densities of noise terms when applying a Kalman smoother, which is a classical statistical tool to forecast time series. More closed to what is proposed in this work, in [20] a time ordered cross-validation is used to validate the training process of ANN when forecasting time series, by testing different number of pattern subsets (or folds), ranging from 2 to 8. Every time a unique fold is used as validation subset, the remaining pattern examples are used to train the ANN. In the next round, another fold is selected for validation and this process is repeated as many times as folds we have. Fig. 1 shows an example of a 5-fold cross-validation.

While one individual (genotype) leads to a single ANN topology (phenotype), applying cross-validation to this individual results in n different ANNs (i.e. different connection weights for the same topology), where n is the number of folds. Given such scheme, there are two relevant issues:

1. How to evaluate the fitness of these n ANNs.

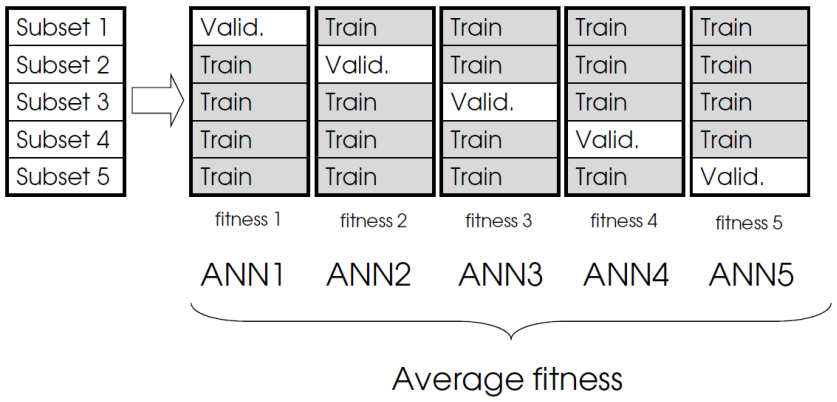

Figure 1: Example of a 5-fold cross-validation.

2. How to combine the outputs of the n different ANNs (explained in Section 2.3).

To measure the fitness of a cross-validation ensemble, the most common procedure is to weight equally each fold fitness, i.e. calculate the fitness value (f_{av}) as the average of all n validation set errors (Fig. 1) [20]:

$$f_{av} = \sum_{i=1}^{n} \frac{1}{n} f_i \tag{2}$$

where f_i denotes the MSE fitness value for the i-th ANN, as measured over the respective validation set (Fig. 1). Yet, in the forecasting domain, recent patterns should have a higher importance when compared with older ones. Following this rational, in this paper, we propose the following weighted cross-validation fitness f_{cv}:

$$\begin{aligned} w_{cvj} &= \begin{cases} 1 - \sum_{i=2}^{n} w_{cvi} & , j = 1 \\ \frac{1}{2^{n+1-j}} & , j \neq 1 \end{cases} \\ f_{cv} &= \sum_{i=1}^{n} w_{cvi} f_i \end{aligned} \tag{3}$$

For example, when $n = 5$, $w_{cv1} = w_{cv2} = 0.0625$, $w_{cv3} = 0.125$, $w_{cv4} = 0.25$ and $w_{cv5} = 0.5$.

2.3. Ensemble combination functions

An ensemble combines several individual models in order to output a single response. Ensembles are often used to improve the predictive perfor-

mance of classification or regression tasks, since aggregated responses tend to perform better than individual ones [21, 22]. Usually, an ensemble is built using variations of the same base learner (e.g. ANN). Unlike most EANN systems, which optimize a single ANN, in this work we propose the use of ADANN to return the best ANN ensemble. Such ensemble is built using the time ordered n-fold cross-validation scheme described in Section 2.2.

To combine the outputs of the n-fold ANN ensemble, simpler solutions are the use of the **mean** or **median** of the n responses. Another alternative is to consider differences between individuals by using a linear combination of the responses, taking into account a weight vector (w_i) that depends on the individual fitnesses:

$$ANNE = \sum_{i=1}^{n} w_i ANN_i \tag{4}$$

where $ANNE$ is the output of the ensemble and ANN_i is the response of the i-th ANN. In this paper, we explore two alternatives to compute the weights: **softmax** (w_{smi}) and **rank-based** (w_{rbi}). The softmax combination works as:

$$\begin{array}{lll} f_i'' & = \frac{f_i' - \min(f')}{\max(f') - \min(f')} & \\ w_{smi} & = \frac{e^{(f_i'')}}{\sum_{k=1}^{n} e^{(f_k'')}} & \text{(softmax function)} \end{array} \tag{5}$$

where $f_i' = 1/f_i$, $\min(f')$ and $\max(f')$ denote the minimum and maximum of all f' values and f_i'' is a scaled transformation of f_i'. The inverse of the validation error (f_i') is used since the lower the MSE the better is the generalization. Also, we compute a soft weight of the scaled f' values, such that $\sum_{i=1}^{n} w_{smi} = 1$.

The rank-based combination function was proposed in [23] and requires more computation. The training series, without test data, is first split (using time order) into two internal training and validation subsets. First, the internal training subset is used as the training data of an initial EANN n-fold validation ensemble run. Then, the internal validation subset is used to tune a β scaling factor. The weighting function assumes that the n ANNs are ranked according to their fitness values (increasing MSE) and the ranked-based weights are computed using:

$$\begin{array}{ll} w_{rbi} & = \frac{e^{(\beta(n+1-i))}}{\sum_{k=1}^{n} e^{(\beta o_k)}} \\ ANNE & = \sum_{i=1}^{n} w_{rbi} ANN_{oi} \end{array} \tag{6}$$

where f_{ok} and ANN_{ok} denote the fitness and output values of the k-th ordered ANN respectively. In this paper, we used a simple hill-climbing method to optimize the β value that provided the lowest error for the internal validation subset. After setting β, another run of the EANN n-fold system is executed, using the whole training series.

3. Experiments and results

3.1. Time series data and forecasting evaluation

We selected six short to medium-sized time series from distinct real-world domains (Table 1) [24]. The series were classified into three groups, according to their seasonal and trended characteristics: the first two are seasonal (with a period of $K = 12$) and trended; the next two contain only a seasonal component ($K = 12$); and the last two are only trended. For all series, we perform from 1 to 19 ahead forecasts (e.g. for paper, the training and test series contain 101 and 19 elements). The global forecasting performance is evaluated the popular Symmetric Mean Absolute Percentage Error (SMAPE) metric [25]:

$$SMAPE = \frac{1}{N} \sum_{i=P+1}^{P+N} \frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2} \times 100\% \quad (7)$$

where P is the index of the last element of the training series and N is the number of forecasts (here, $N = 19$). When compared with MSE, SMAPE has the advantage of being scale independent and less sensitive to outliers. The SMAPE values range from 0% to 200% and low values suggest a high quality model.

Table 1: Time series data.

Series	K	**Tr.**	**Size**	**Description (region, years)**
paper	12	Yes	120	Monthly sales of paper (France, 1963-72)
pass.	12	Yes	144	Monthly int. airline passengers (1949-60)
ozone	12	No	180	Ozon concentration (Azusa, 1956-70)
temperature	12	No	240	Monthly air temp. (Nottingham, 1920-39)
Dow-Jones	–	Yes	157	Dow-Jones index monthly closings (1968-81)
IBM	–	Yes	369	Daily IBM closing stock prices (1961-1962)

3.2. Cross-validation ensemble results

For each time series, we applied the ADANN system using cross-validation with (f_{cv}) and without (f_{av}) weighted fitnesses. The number of folds (or subsets) was ranged from 2 to 8. For both cross-validation fitness schemes, we computed the four ensemble combination methods described in Section 2.3: **mean**, **median**, **softmax** and **rank-based**. Since a large number of experiments was conducted and due to space limitations, only aggregated results are presented for all ensemble combinations. Table 2 shows the average SMAPE value (over all six series) for the distinct ANN ensemble strategies and number of folds. For benchmark purposes, we also tested the no cross-validation scheme (0 folds), which uses the simpler time order holdout split, where the validation set contains the most recent 30% elements of the whole training series and the remaining data is used to fit the model.

Each ensemble combination is evaluated by its average value over all 2 to 8 folds ($\overline{\text{Folds}}$) and also according to the number of wins against the 0 fold benchmark. In Table 2, **bold** in column Folds denotes better than the 0 fold benchmark, while in column $\overline{\text{Folds}}$ denotes the best value. The two cases where SMAPE=7.3 (e.g. last row, 5-folds) are in bold (i.e. better than the benchmark) since the table presents round values (e.g. 7.28). An analysis to Table 2 shows that weighted cross-validation f_{cv} outperforms the averaging method f_{av} for computing the ensemble fitness. For all four combination functions, the $\overline{\text{Folds}}$ value is always lower for f_{cv} when compared with f_{av}. Also, the weighted fitnesses present more fold setups that outperform the 0 fold benchmark. Turning to the combination function comparison, the **rank-based** combination is the best option for both fitness calculation methods, followed by the **median**. Overall, the best ensemble is given by the f_{cv} and **rank-based** combination. Such combination achieves the smallest $\overline{\text{Folds}}$ value and also presents 5 wins against the benchmark. Moreover, the non parametric Mann-Whitney test was applied to compare the f_{cv} and **rank-based** combination against the remaining methods, showing a statistical significance (p-value$<$0.05) in all comparisons except for the average **rank-based** ensemble (Table 2).

The full SMAPE results (i.e. for all time series) for the best ensemble strategy (f_{cv}, **ranked-based**) are presented in Table 3. In the table, underline denotes the best value for each row and **bold** denotes values that outperform the baseline (0 folds). The last row presents the average over all series (the same row values that appear in Table 2). The cross-validation ANN ensemble outperforms the benchmark in almost all series (the excep-

Table 2: Forecasting results (average % SMAPE).

ANN Ensemble Strategy	Folds 0	2	3	4	5	6	7	8	$\overline{\text{Folds}}$
f_{av}, **mean**	7.3	9.3	8.5	8.1	7.6	8.2	**6.9**	8.8	8.2†
f_{av}, **median**	7.3	9.3	8.3	7.9	7.5	7.9	**7.1**	7.5	7.9†
f_{av}, **softmax**	7.3	9.4	8.4	8.4	7.4	8.1	**6.9**	8.8	8.2†
f_{av}, **rank-based**	7.3	8.7	7.9	7.4	**7.1**	7.4	**6.6**	7.8	7.5
f_{cv}, **mean**	7.3	8.4	8.7	**7.1**	8.1	7.8	**7.3**	**7.1**	7.8†
f_{cv}, **median**	7.3	8.4	8.4	**7.2**	8.0	7.7	**7.2**	**7.2**	7.7†
f_{cv}, **softmax**	7.3	9.4	8.8	**6.9**	8.5	7.8	**7.1**	**7.3**	8.0†
f_{cv}, **rank-based**	7.3	8.0	8.1	**6.4**	**7.3**	**6.8**	**6.5**	**6.4**	**7.0**

† - significantly worst than f_{cv} and **rank-based**, under a Mann-Whitney test.

tion is passengers). In particular, in series ozone, temperature and IBM, all 2 to 8 fold setups provide better forecasts when compared with the simpler holdout ANN method.

Regarding the best number of folds (n), it is dependent on the time series considered. For instance, in Table 3 using 4 folds is the best option for paper and Dow-Jones series, while the 8-folds method gets the best results for the temperature and IBM data. For f_{cv} and **ranked-based** (Table 2), the 4-fold setup obtains the second best overall performance: $\overline{\text{SMAPE}}$ =6.37%, which is only slightly higher than $\overline{\text{SMAPE}}$ =6.35% for the 8-fold ensemble. Turning to the computational complexity, the higher the number of folds used, the higher is the computational effort required. For example, for IBM series (the largest dataset), f_{cv} and **ranked-based** ensemble, and under the same computer processor, running the EANN for 0-fold required 1140 minutes, a value that increased to 2004 minutes for 4-fold and 6875 minutes for 8-fold. Given this trade-off, we suggest the 4-fold setup as a reasonable balance between accuracy and computational effort. Under the Mann-Whitney test, the 4-fold ensemble is significantly better (p-value<0.05) than the 2, 3 and 6-fold variants. When compared against the 0-fold setup, the test does not show statistical significance of the 4-fold (p-value of 0.22) and 8-fold (p-value of 0.16) methods. Nevertheless, the respective p-values are much lower than the ones obtained in the remaining comparisons (e.g. p-value of 0.84

for 4-fold vs 8-fold). For demonstration purposes, Table 4 shows the best models' parameters (e.g. number of input nodes) evolved by ADANN for the **ranked-based** 4-fold and weighted fitness cross-validation method.

Table 3: Best ANN ensemble (f_{cv}, **ranked-based**) forecasting results (%SMAPE values).

Time	Folds							
Series	0	2	3	4	5	6	7	8
paper	8.2	**7.9**	8.4	**7.5**	**7.8**	**7.5**	**7.5**	8.8
passengers	3.2	11.4	10.7	5.1	10.2	5.5	4.1	3.4
ozone	16.6	**15.3**	**16.1**	**15.2**	**13.8**	**15.6**	**15.7**	**15.4**
temp.	4.3	**3.7**	**3.7**	**3.6**	**3.7**	**3.8**	**3.7**	**3.5**
Dow-Jones	6.7	7.0	7.0	**4.5**	**4.9**	**6.1**	**5.6**	**4.9**
IBM	5.1	**2.8**	**2.5**	**2.4**	**3.3**	**2.4**	**2.2**	**2.1**
Average	7.3	8.0†	8.1†	**6.4**	**7.3**	**6.8**†	**6.5**	**6.4**

† - significantly worst than 4-fold, under a Mann-Whitney test.

Table 4: Best ANN structures and learning rate evolved by ADANN for 4-folds and f_{cv}.

Time Series	**Input Nodes**	**Hidden Nodes**	**Learning Rate**
paper	37	67	0.19
passengers	50	48	0.52
ozone	48	52	0.63
temperature	75	58	0.42
Dow-Jones	45	98	0.56
IBM	10	20	0.92

3.3. Comparison with exponential smoothing

As a baseline comparison, we adopted the Holt-Winters method, from the family of exponential smoothing methods and often used to predict series with trended and seasonal factors [6]. The predictive model is based on underlying patterns, such as trends and seasonality, that are distinguished

from random noise by averaging the historical values [7]. Its popularity is due to advantages such as reduced computational demand, simplicity of use and accuracy of the forecasts, specially with seasonal series. In this paper, we tested the multiplicative seasonal Holt-Winters model (with 3 internal parameters) for the seasonal series, with a period of $K = 12$, while for the non seasonal series (Dow-Jones and IBM) we used the exponential with trend version (with 2 internal parameters). To optimize the Holt-Winters internal parameters, we adopt a 0.05 grid search for the best training error (MSE), which is a common procedure within the forecasting field.

For the comparison, we selected the setup proposed in the Section 3.2: the weighted and rank-based $4 - fold$ ANN ensemble. The forecasting results are presented in Table 5 and show a competitive performance of the ensemble. While the average SMAPE value is identical for both methods, the ensemble outperforms Holt-Winters in 4 of the 6 tested series, namely paper, temperature, Dow-Jones and IBM. This is an interesting outcome, as the Holt-Winters was specifically developed for series with seasonal and/or trended components. Furthermore, the ensemble approach does not use any a priori knowledge (e.g. the model does not know that seasonal series are monthly ones). Hence, if computation power is available, our proposed ensemble can be a valuable forecasting alternative for short or medium-sized time series with seasonal or trended components.

Table 5: Comparison of the suggested approach versus Holt-Winters (SMAPE%).

Time Series	**Ensemble**	**Holt-Winters**
paper	**7.5**	8.4
passengers	5.1	**3.0**
ozone	15.2	**12.3**
temperature	**3.6**	**3.6**
Dow-Jones	**4.5**	7.7
IBM	**2.3**	3.1
Average	**6.4**	**6.4**

4. Conclusions

In this paper, we proposed an evolutionary artificial neural network engine to evolve a fitness weighted n-fold cross-validation artificial neural network ensemble scheme for time series forecasting. To combine the n ANN outputs into a single response, we explored four distinct combination functions. Experiments held with six time series, with different characteristics and from different domains. As the main outcome of this work, we show that the fitness weighted n-fold ensemble improves the accuracy of the forecasts, outperforming both the no weight $n-$fold ensemble and the simpler holdout validation (0-fold) EANN. Also, as a compromise between accuracy and computational cost, based on the presented results, we advise the use of a 4-fold ANN ensemble that is evolved using weighted cross-validation and that uses a rank-based combination method to build the final forecasts. Moreover, when compared with a classical method like Holt-Winters, competitive forecasting results were achieved by the proposed approach, showing that it can be an interesting alternative. In future work, we intend to use the EANN engine to evolve ensembles of sparsely connected ANNs [10]. We also intend to apply a similar approach to evolve ensembles of other base learners, such as support vector machines [26].

Acknowledgements

This work was supported by University Carlos III of Madrid and by Community of Madrid under project CCG10-UC3M/TIC-5174. The work of P. Cortez was funded by FEDER (program COMPETE and FCT) under project FCOMP-01-0124-FEDER-022674.

[1] A. Abraham, Editorial - hybrid soft computing and applications, International Journal of Computational Intelligence and Applications 8 (2009).

[2] J. Sedano, L. Curiel, E. Corchado, E. de la Cal, J. R. Villar, A soft computing method for detecting lifetime building thermal insulation failures, Integrated Computer-Aided Engineering 17 (2010) 103–115.

[3] S. García, J. Derrac, J. Luengo, C. J. Carmona, F. Herrera, Evolutionary selection of hyperrectangles in nested generalized exemplar learning, Applied Soft Computing 11 (2011) 3032–3045.

[4] E. Corchado, A. Herrero, Neural visualization of network traffic data for intrusion detection, Applied Soft Computing 11 (2011) 2042–2056.

[5] T. Wilk, M. Wozniak, Soft computing methods applied to combination of one-class classifiers, Neurocomputing 75 (2012) 185–193.

[6] S. Makridakis, S. Wheelwright, R. Hyndman, Forecasting methods and applications, 3rd Edition, John Wiley & Sons, USA, 2008.

[7] P. R. Winters, Forecasting sales by exponentially weighted moving averages, Management Science 6 (1960) 324–342.

[8] M. Štěpnička, A. Dvořák, V. Pavliska, L. Vavříčková, A linguistic approach to time series modeling with the help of f-transform, Fuzzy Sets and Systems 180 (1) (2011) 164 – 184.

[9] G. Zhang, B. Eddy Patuwo, M. Y. Hu, Forecasting with artificial neural networks: The state of the art, International Journal of Forecasting 14 (1) (1998) 35–62.

[10] P. Cortez, M. Rocha, J. Neves, Time Series Forecasting by Evolutionary Neural Networks, Idea Group Publishing, USA, 2006, Ch. III, pp. 47–70.

[11] J. Peralta, X. Li, G. Sanchez, A. Sanchis, Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm, Neural Computing & Applications.doi:10.1007/s00521-011-0741-0.

[12] X. Yao, A review of evolutionary artificial neural networks, International Journal of Intelligent Systems 8 (4) (1993) 539–567.

[13] A. Abraham, Meta learning evolutionary artificial neural networks, Neurocomputing 56 (2004) 1–38.

[14] M. Rocha, P. Cortez, J. Neves, Evolution of neural networks for classification and regression, Neurocomputing 70 (2007) 2809–2816.

[15] D. B. Fogel, Evolutionary computation: toward a new philosophy of machine intelligence, IEEE Press, Piscataway, NJ, USA, 1995.

[16] J. Peralta, G. Gutierrez, A. Sanchis, Adann: automatic design of artificial neural networks, in: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation, GECCO '08, ACM, NY, USA, 2008, pp. 1863–1870.

[17] L. Prechelt, Early stopping-but when?, in: G. B. Orr, K.-R. Müller (Eds.), Neural Networks: Tricks of the Trade, outgrowth of a 1996 NIPS workshop, Springer, 1998, pp. 55–69.

[18] F. Mosteller, A k-sample slippage test for an extreme population, The Annals of Mathematical Statistics 19 (1) (1948) 58–65.

[19] S. Sarkka, A. Vehtari, J. Lampinen, Cats benchmark time series prediction by kalman smoother with cross-validated noise density, Neurocomputing 70 (13-15) (2007) 2331 – 2341.

[20] B. Wah, M. Qian, Time-series predictions using constrained formulations for neural-network training and cross validation, in: Proc. Intl Conf. on Intelligent Information Processing, 16th IFIP World Computer Congress, 2000, pp. 220–226.

[21] S. Geman, E. Bienenstock, R. Doursat, Neural networks and the bias/variance dilemma, Neural Computation 4 (1992) 1–58.

[22] A. Krogh, P. Sollich, Statistical mechanics of ensemble learning, Physical Review E 55 (1) (1997) 811–825.

[23] X. Yao, M. Islam, Evolving artificial neural network ensembles, Computational Intelligence Magazine 3 (1) (2008) 31–42.

[24] R. Hyndman, Time series data library, http://robjhyndman.com/tsdl/ (accessed June, 2011)).

[25] R. Hyndman, A. Koehler, Another look at measures of forecast accuracy, International Journal of Forecasting 22 (4) (2006) 679–688.

[26] P. Cortez, Sensitivity Analysis for Time Lag Selection to Forecast Seasonal Time Series using Neural Networks and Support Vector Machines, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN 2010), IEEE, Barcelona, Spain, 2010, pp. 3694–3701.