Neurocomputing 491 (2022) 509–521

Contents lists available at ScienceDirect

Neurocomputing

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

Strategies for time series forecasting with generalized regression neural networks

Francisco Martínez [a,*], Francisco Charte [a], María Pilar Frías [b], Ana María Martínez-Rodríguez [b]

[a] Department of Computer Science, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Jaén, Jaén 23071, Spain
[b] Department of Statistics and Operations Research, University of Jaén, Jaén 23071, Spain

ARTICLE INFO

Article history:
Received 17 March 2021
Revised 21 July 2021
Accepted 12 December 2021
Available online 24 December 2021

Keywords:
Generalized regression neural networks
Time series forecasting
Software

ABSTRACT

This paper discusses how to forecast time series using generalized regression neural networks. The main goal is to take advantage of their inherent properties to generate fast, highly accurate forecasts. To this end, the key modeling decisions involved in forecasting with generalized regression neural networks are described. To deal with every modeling decision, several strategies are proposed. Each strategy is analyzed in terms of forecast accuracy and computational time. Apart from the modeling decisions, any successful time series forecasting methodology has to be able to capture the seasonal and trend patterns found in a time series. In this regard, some clever techniques to cope with these patterns are also suggested. The proposed methodology is able to forecast time series in an automatic way. Additionally, the paper introduces a publicly available R package that incorporates the best presented modeling approaches and transformations to forecast time series with generalized regression neural networks.

1. Introduction

Time series forecasting is a common task in many fields such as business [1], energy [2] or environment [3]. Effective forecasts can save a lot of time and money, facilitating planning, scheduling and other activities.

There are situations in which a great number of time series need to be forecast quickly, such as the sells of different retail products. In this context fast forecasting tools, which can be applied in an automatic way, are highly valuable. These requirements preclude some common techniques; for example, the ARIMA methodology [4] is usually applied under the supervision of an expert, while other forecasting tools can be used automatically but have high computational needs [5].

Generalized regression neural networks (GRNNs) [6], a variant of radial basis function networks [7], exhibit interesting properties to develop a fast forecasting tool: 1) they have a single-pass learning, 2) they only need to set or fit one parameter and 3) they produce deterministic results, so that it is not necessary to train several neural networks to achieve more accurate and trustworthy results.

Motivated by these properties, this paper analyzes how to forecast time series using generalized regression neural networks in an effective way. Several strategies to deal with the key factors involved in forecasting time series with generalized regression neural networks are proposed. These insights and strategies should assist the forecaster in finding a balance between automatic, fast predictions and highly accurate forecasts.

All the techniques explained in this paper have been incorporated into an R package that is publicly available on CRAN, the Comprehensive R Archive Network. This way, any practitioner can easily benefit from the different approaches proposed in this work.

The remainder of this paper is structured as follows. In Section 2 generalized regression neural networks are analyzed. Also, it is discussed how these neural networks can be applied in a time series forecasting context. In Section 3 several modeling alternatives are proposed. Section 4 explains how to transform the training examples so that GRNN can accurately forecast a time series with a trend. In Sections 3 and 4 different strategies, applied in previous works, are also described. In Section 5 the proposed strategies are analyzed according to forecast accuracy and computational time using two data sets. Section 6 introduces the `tsfgrnn` R package, which incorporates the best modeling and transformation approaches proposed in this work. Finally, Section 7 draws some conclusions.

* Corresponding author.
E-mail addresses: fmartin@ujaen.es (F. Martínez), fcharte@ujaen.es (F. Charte), mpfrias@ujaen.es (M.P. Frías), ammartin@ujaen.es (A.M. Martínez-Rodríguez).

https://doi.org/10.1016/j.neucom.2021.12.028
0925-2312/

2. Time series forecasting with GRNN

A generalized regression neural network is a variation of a radial basis neural network proposed by Specht [6] and used primarily for classification and regression. A GRNN is an artificial neural network made up of three layers: the input, hidden and output layer, see Fig. 1. The hidden layer has radial basis neurons whose centers are the training examples. Normally, the radial basis function is the multivariate Gaussian function:

$$G(x, x_i) = \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right) \tag{1}$$

where x_i and σ are the center and the smoothing parameter respectively and x is the input vector. The output of a hidden layer neuron is related to the closeness of the input vector to the center, scaled by the smoothing parameter.

Given a training set consisting of n training patterns and their associated targets—vectors $\{x_1, x_2, \ldots x_n\}$ and $\{y_1, y_2, \ldots y_n\}$ respectively—, the output for an input pattern x is computed in two steps. First, the hidden layer produces a set of weights associated with the closeness of x to the training patterns:

$$w_i = \frac{\exp\left(-\frac{\|x-x_i\|^2}{2\sigma^2}\right)}{\sum_{j=1}^{n} \exp\left(-\frac{\|x-x_j\|^2}{2\sigma^2}\right)} \tag{2}$$

The weights sum to one and represent the contribution of every training pattern to the final result. Next, the output layer computes the output as:

$$\hat{y} = \sum_{i=1}^{n} w_i y_i \tag{3}$$

so a weighted average of the training targets is obtained, where the weights are related to the closeness of the input to the training patterns—the closer the higher. The role of the smoothing parameter is to control how many targets have significant weights in the weighted average, that is, the level of smoothing in the output. When σ is large all of the targets have a small and similar weight, so the result is close to the mean of the targets. On the other hand, when σ is small only the targets whose patterns are close to the input have significant weights.

2.1. Application of GRNN regression to time series forecasting

In order to apply GRNN regression in a time series forecasting setting the historical values of a series are used to extract a set of examples—a training set. An example is a pair formed by a target pattern, a historical value of the series, and a training pattern consisting of several historical values previous to the target pattern. This way an example relates the future behavior of a series—target pattern—with its past behavior—training pattern. For instance, given the time series $s = \{y_1, y_2, \ldots, y_{40}\}$ and assuming that an example is formed by a historical value representing the target pattern and a training pattern consisting of the four previous historical values of the target, a subset of the 36 examples that can be extracted from s are shown in Fig. 2.

The set of examples is used to train the GRNN model, so it can learn how to predict the future behavior from its past. Once trained, the input pattern to the model are the last historical values of the series, its last behavior, which is used to predict the future values of the series. The input pattern has the same structure that the training patterns of the examples. For instance, for the previous example, whose training set is shown in Fig. 2, the input to the model would be the vector $\{y_{37}, y_{38}, y_{39}, y_{40}\}$, that is, the last four values of the series.

As explained before, to generate its prediction a GRNN model produces a weighted average of the target patterns of its training set. In this weighted average, the weight of each target is related to the closeness between its associated training pattern and the input pattern. For instance, suppose that the next future value of the time series in Fig. 3, consisting of 40 historical values, needs to be forecast. For this purpose a GRNN model is trained. Each example in the training set is formed by a historical value, the target pattern, and a training pattern consisting of the four previous historical values of its target. For the time series in Fig. 3, Fig. 4 shows its first training example and the input pattern used to predict the next future value of the series. The training pattern of this first example is relatively close to the input pattern so its target pattern will have a significant weight. On the other hand, the training pattern of the second example—see Fig. 5—is not so close to the input pattern and therefore its associated target will have a smaller weight.

The intuition behind using GRNN for time series forecasting is that previous patterns that are close—similar—to the latest values of the series can be found in the hope that a smoothed combination of their subsequent patterns will be similar to the future behavior of the series.

In spite of their interesting properties, GRNN has not been commonly used to forecast time series. In [8], the forecast accuracy of several machine learning models, including GRNN, are compared using a subset of the monthly time series from the M3 competition [9]. The smoothing parameter is selected among a number of preset values using K-fold cross-validation. In our opinion this comparison has important limitations, first and foremost, only one step ahead forecasts are considered. Also, one of the contestants of the NN3 competition [10] used GRNN, achieving the best predictions for the reduced data set, but a modest rank for the whole data set, a version of the algorithm is presented in [11]. In [2], several neural networks approaches are

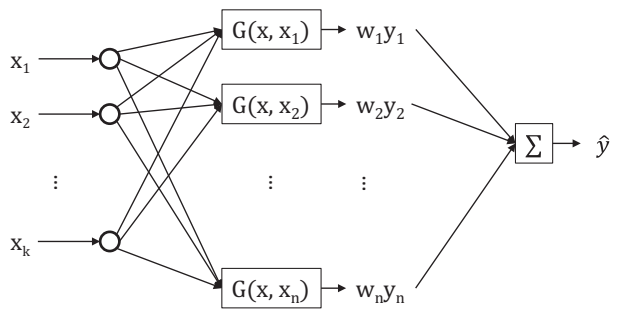

Fig. 1. Structure of a GRNN.

	Training pattern	Target pattern
Example 1	y_1, y_2, y_3, y_4	y_5
Example 2	y_2, y_3, y_4, y_5	y_6
Example 3	y_3, y_4, y_5, y_6	y_7
	...	...
Example 36	$y_{36}, y_{37}, y_{38}, y_{39}$	y_{40}

Fig. 2. Training set from series $\{y_1, y_2, \ldots, y_{40}\}$ when a training pattern consists of the four previous values of its target pattern.

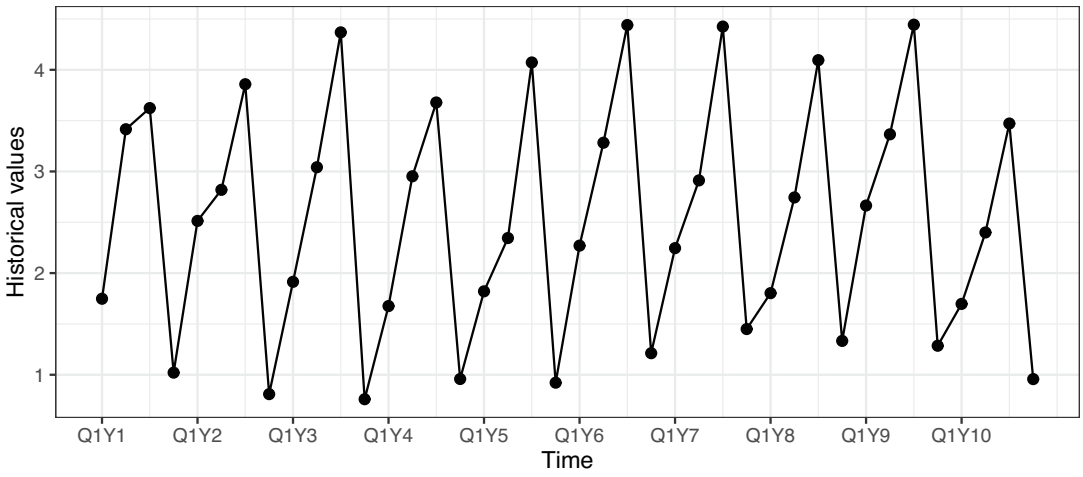

Fig. 3. Time series consisting of 40 historical values.

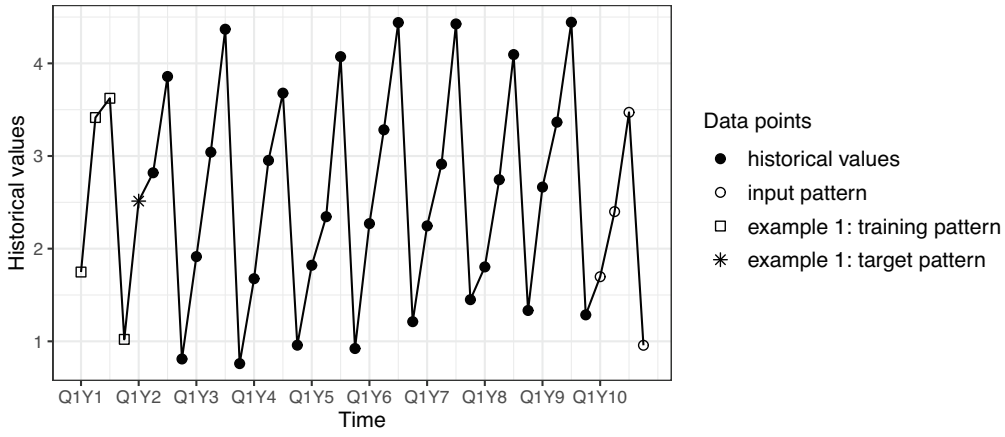

Fig. 4. Forecasting with GRNN: first example of the training set and input pattern.

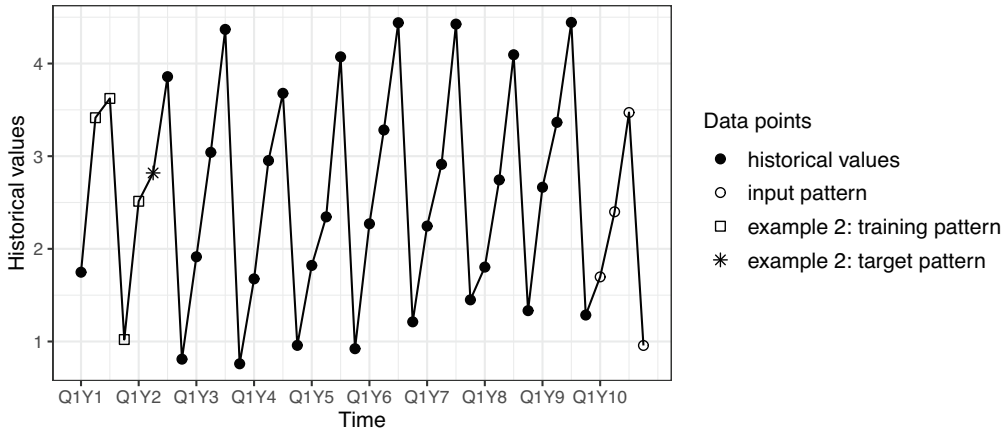

Fig. 5. Forecasting with GRNN: second example of the training set and input pattern.

compared for short-term load forecasting, the best results being achieved by GRNN. In [12], time series forecasting tools available in CRAN that can be applied automatically are analyzed; among the machine learning approaches a preliminary version of the package explained in Section 6 obtains the best results in terms of forecast accuracy.

3. GRNN modeling

In order to use GRNN in a time series forecasting context several modeling decisions have to be made. This section discusses some alternative modeling strategies. The implications of every strategy with regard to speed and forecast accuracy are analyzed.

3.1. The smoothing parameter

The smoothing parameter, σ, plays a key role in the way a time series is forecast. The larger σ the more targets with significant weights in the output of the GRNN. The optimal value of σ depends on the series. For example, Fig. 6 shows a time series generated by the following equation:

$$y_t = 10 + \epsilon_t \tag{4}$$

where ϵ_t represents i.i.d. random noise. Fig. 6 also shows two predictions for the next 5 future values of the series. In this case the model with a large σ is better, because all the targets have a similar weight and therefore the forecasts are close to the mean of the historical values. On the other hand, the quarterly time series in Fig. 3 has a strong seasonal component. If the quarters $\{Q1, Q2, Q3, Q4\}$ of the next year have to be forecast, it would be desirable that the targets with significant weights have quarters $\{Q1, Q2, Q3, Q4\}$ and not any other combination, such as $\{Q2, Q3, Q4, Q1\}$. Section 3.3 explains how to achieve this. Fig. 7 shows how the use of two different smoothing parameters affects the forecast of the four quarters of the next year. When σ is very large the forecasts are close to the mean of the historical values. However, when σ is small the significant targets are $\{Q1, Q2, Q3, Q4\}$ patterns and the forecast captures the seasonal behavior.

As noted above, a right choice of the smoothing parameter is crucial for forecast accuracy. Therefore, we propose that the value of σ is chosen automatically using an optimization tool that finds the value of σ that minimizes a forecasting accuracy measure on a validation set formed by the last h historical values of the time series, where h is the forecasting horizon. The *forecasting horizon* is the number of future values to be forecast, for example, if the forecasting horizon is 3, then the next 3 future values of the series are predicted. To assess the forecast accuracy, when optimizing σ, rolling origin evaluation can be applied. Rolling origin evaluation [13] is an iterative technique that tries to get the most out of the validation set. Let us see how rolling origin works when the length of the validation set is h. First, rolling origin assesses a test set formed by the last h historical values using a training set consisting of the previous historical values. This procedure is repeated $h-1$ more times, moving the origin of the test set one value ahead each time, see Fig. 8. This way, h one step ahead predictions can be assessed, $h-1$ two steps ahead predictions, and so on. This is a great contrast with the fixed origin evaluation that, for a validation set of size h, is only able to assess one prediction for each one of the h horizons. However, rolling origin evaluation is more computationally intensive than fixed origin evaluation.

Previous works in time series forecasting using GRNN propose other ways of choosing the smoothing parameter. For example, in [8] σ is chosen from the possible values [0.05, 0.1, 0.2, 0.3, 0.5, 0.6, 0.7] using K-fold validation. This method can be faster than our proposal, but we consider it somewhat arbitrary to restrict the values of such a critical parameter to this small set of possible values. In [2] a quite sophisticated method is used that, we must admit, we do not fully understand. In [11], in order to avoid a time consuming selection method associated with optimizing σ, three different GRNN models are used with different smoothing parameters. The final forecast is the average of the forecasts of the three models.

3.2. Multiple step ahead strategy

So far it has been assumed that when the forecasting horizon is h, the target patterns are vectors of h consecutive historical values and GRNN outputs a weighted combination of these targets. This is sometimes called the MIMO—Multiple-Input Multiple-Output—strategy for forecasting multiple steps ahead values, see [14]. The MIMO approach is used in [2] in a context of time series forecasting with GRNN. However, other approaches to forecast multiple values exist [14]. In the iterative or recursive approach, used in classical time series forecasting methodologies such as ARIMA or exponential smoothing [15], the forecast function generates only one step ahead forecasts—using GRNN the target patterns are length-one vectors. To predict h values the forecast function is used h times in an iterative way, previous forecasts are used as input when historical observations are not available.

Some works in time series forecasting with GRNN apply other approaches. For example, in [11] the direct strategy [14] is used. This strategy builds a different model to forecast each future value. That is, if the forecasting horizon is h, h different models are built and each model predicts a different future value.

3.3. Feature selection

In GRNN a training example consists of a pair of vectors with historical values: a target pattern vector and a training pattern vector with previous values of the target. The length of a target vector is determined by the multiple step ahead strategy: with the recur-

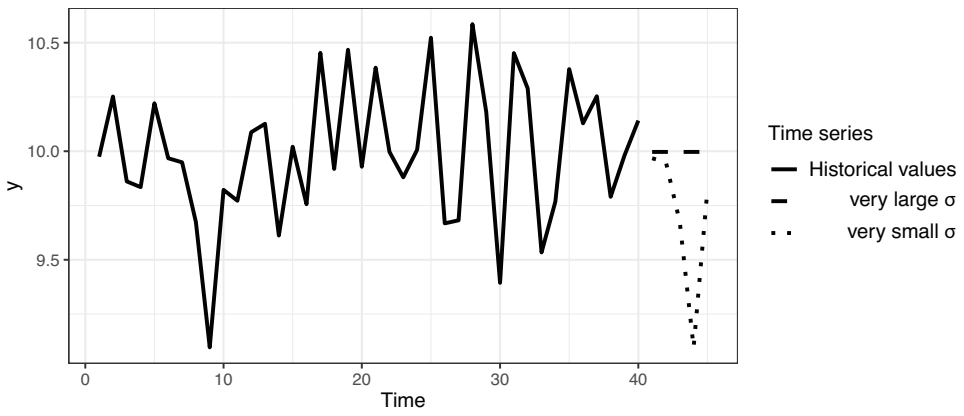

Fig. 6. Effect of smoothing parameter in forecasting stationary series.

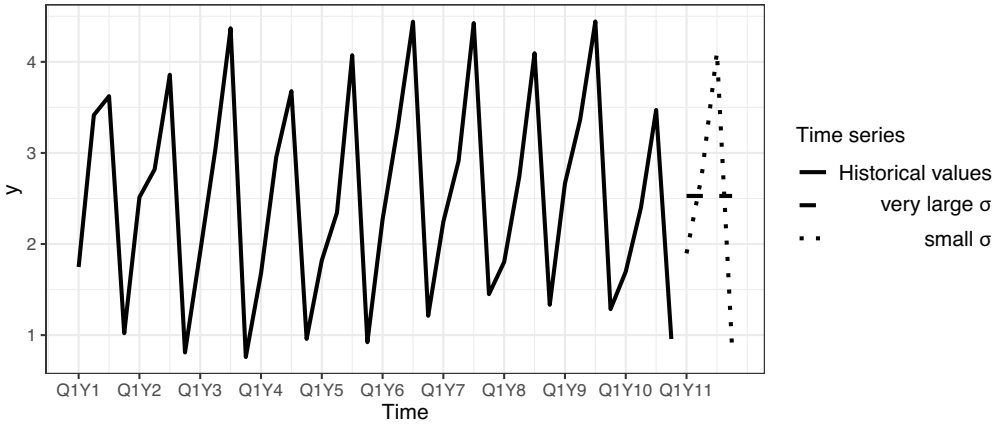

Fig. 7. Effect of smoothing parameter in forecasting seasonal series.

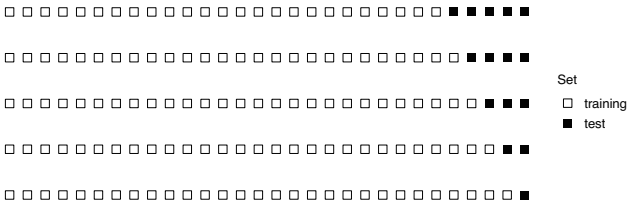

Fig. 8. Training and test sets with rolling origin evaluation when $h = 5$.

sive and direct approach the length is one, with MIMO is h, where h is the forecasting horizon. On the other hand, the lags used to create the training patterns have to be selected. Next, we propose a straightforward heuristic to select the lagged values used as training patterns. The terms *lag* and *lagged value* are used to indicate a previous value of a historical value of a series. For example, given a historical value, its lags or lagged values 1 to 4 are its four previous values in the series.

The proposed heuristic works as follows:

- For seasonal time series—weekly, monthly, quarterly, ...—the training patterns are lagged values from 1 to s, where s is the length of the seasonal period, that is, 7 for weekly series, 12 for monthly series, etc. These lagged values are chosen because they help to capture a seasonal behavior. For example, in Fig. 9 a quarterly series with a seasonal pattern is shown, the first three quarters of a year have a similar level, quite higher than the last quarter. If the lagged values 1 and 2 are used to generate the training patterns, then a wrong forecast will be obtained. To see why, suppose that the fourth quarter of the current year is predicted. In Fig. 9 two of the closest training patterns to the input are highlighted. Their targets will have, therefore, significant weights. Unfortunately, the target of the first training pattern is a third quarter value, an undesirable scenario because a fourth quarter is predicted. However, if lagged values 1 to 4 were used, the closest training patterns would have a fourth quarter as target—Fig. 10 shows two of the closest training patterns to the input when lagged values 1 to 4 are used.
- If the time series is not seasonal and the partial autocorrelation function—PACF—has significant lags, then these significant lags are selected to build the training patterns. Although the PACF only tests for linear relationship, this way of selecting lagged values seems to produce good results [16].
- Finally, if the time series is not seasonal and its PACF has no significant lags, lagged values 1 to 5 are selected.

In order to determine whether a time series is seasonal or not, the seasonal test used by the benchmark methods of the M4 competition [17] has been applied. Given a series of n values and frequency s—s is 1 for yearly data, 4 for quarterly data, 12 for monthly data and so on—the test is based on the autocorrelation

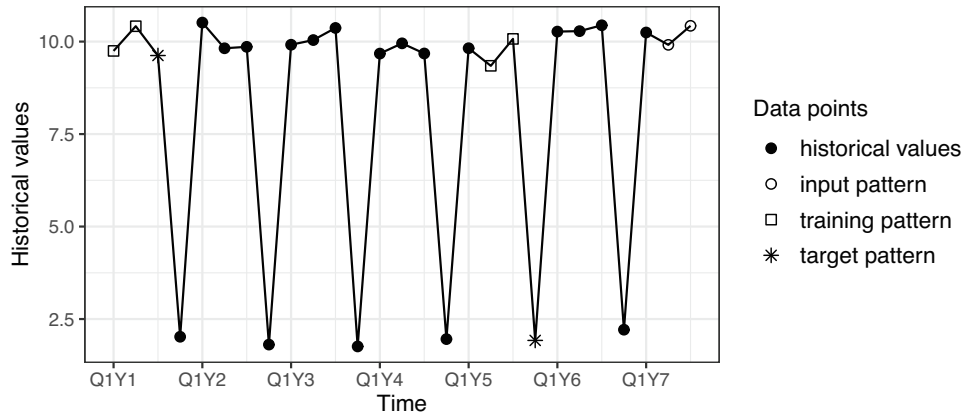

Fig. 9. Quarterly series with strong seasonality and lagged values 1 and 2.

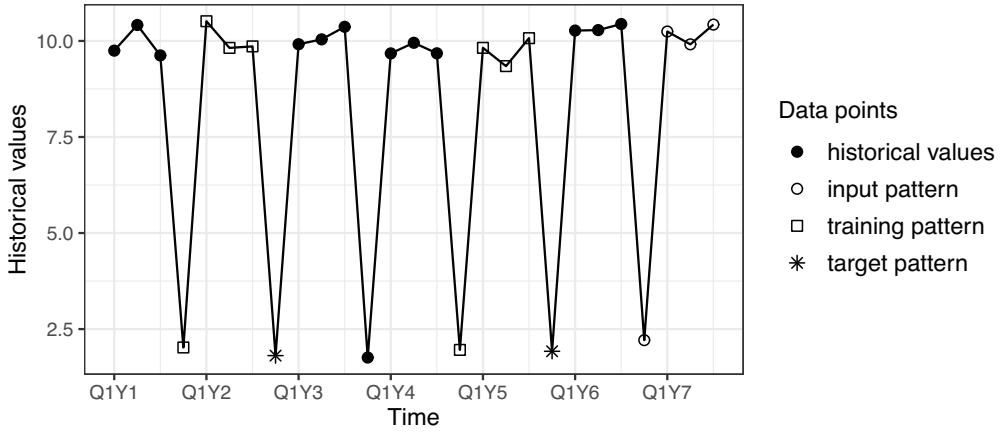

Fig. 10. Quarterly series with strong seasonality and lagged values 1 to 4.

function. The series is considered seasonal if the following rule is fulfilled:

$$|ACF_s| > 1.645\sqrt{\frac{1+2(ACF_1+\sum_{i=2}^{s-1}ACF_i^2)}{n}} \quad (5)$$

where ACF_i is the value of the autocorrelation function for lag i. Non-seasonal series ($s = 1$) and series where $n < 3s$ are not tested and assumed as not being seasonal.

Previous works in time series forecasting with GRNN apply other strategies for feature selection. For example, in [8] K-fold cross-validation is used to select among the following sets of lagged values: lag 1, lags 1 to 2, lags 1 to 3, lags 1 to 4 and lags 1 to 5. In [11] a similar strategy is applied, but the set of possible lagged values is extended until lags 1 to 12. On the other hand, in [2] hourly electricity demand is predicted and knowledge of the behavior of this kind of series allows the author to select lags 1 to 24 in order to capture the hourly demand pattern.

4. Transformations

Time series forecasting methodologies often transform or preprocess time series to improve forecast accuracy. Most of these transformations are aimed at dealing with seasonal and trend patterns. This section explains our proposal to cope with these patterns.

4.1. Trend

Clearly, GRNN is not suited to predict a time series with a trend. This is due to the fact that GRNN predicts a combination of historical values and the range of these values is probably different from the range of future values. For example, in Fig. 11 a time series with an additive trend is shown. The series has been generated by the following equation:

$$y_t = 1.5t + \epsilon_t$$

where ϵ_t represents i.i.d. random noise. Fig. 11 also shows the forecast of GRNN with the recursive multiple step ahead strategy. The smoothing parameter is chosen automatically: a small value has been selected so that only the latest historical values have significant weights. Despite this, it is highly likely that these values are out of the range of the actual future values.

In order to forecast a time series with a trend the following transformation, called additive transformation, is proposed:

- Given a training example consisting of a training pattern vector and a target pattern vector, the target vector is transformed by subtracting the mean of the training pattern vector. This way, a prediction is a weighted combination of transformed targets. To back transform a prediction, the mean of the input vector is added to it. Fig. 12 shows how the time series in Fig. 11 is predicted when this transformation has been applied. Note that with this transformation a target represents the difference of the target to the mean level of its training pattern. In Fig. 12 the forecast using a multiplicative transformation, which is explained below, is also shown.

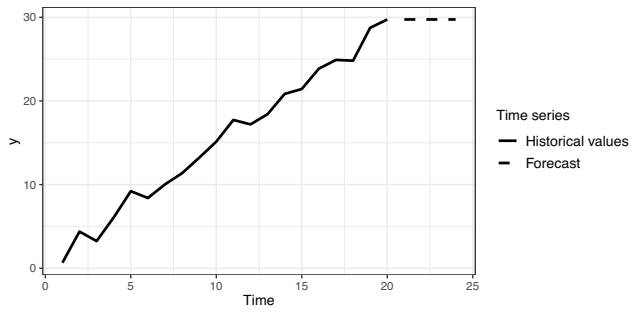

Fig. 11. GRNN is not suited to forecast a series with a trend.

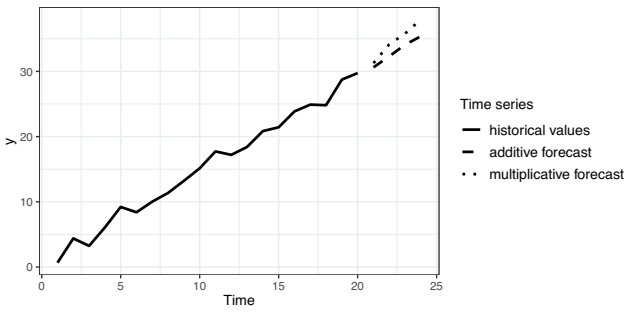

Fig. 12. Forecasting with transformed training examples.

Apart from transforming the target pattern vectors, we also suggest to transform the training pattern vectors. Each training pattern vector—and the input vector—is transformed by subtracting its mean. The goal of this transformation is that an input vector can be considered close to any training pattern vector with a similar shape, irrespective of the level of the input and training pattern vector. For example, Fig. 13 shows a time series with a level shift in which the input and two training patterns are highlighted. The two training patterns are quite similar in shape to the input. However, the patterns have a very different mean level, so the second one is considered quite closer than the first one taking into account that closeness is implemented by using the Euclidean distance. Fortunately, with the aforementioned transformation both training patterns will be similar. This transformation is inspired by the way in which time series are clustered by their shape [18].

Fig. 14 shows an example of transforming the training examples of time series $\{1, 3, 6, 7, 2, 9, 5\}$, when the training targets are two consecutive historical values and the training patterns two lagged values of the targets.

Besides this transformation, we alternatively suggest a multiplicative transformation in which the value of a training example is divided by the mean of its training pattern and a prediction is back transformed by multiplying it by the mean of the input. The additive transformation is suited to forecast series with additive trends, such as the one in Fig. 12, in which the multiplicative transformation overestimates the trend. On the other hand, time series with a multiplicative trend are common in economics. For instance, the following equation:

$$y_t = 10 \cdot 1.05^t + \epsilon_t \tag{6}$$

where ϵ_t represents i.i.d. random noise, generates a series with a multiplicative trend. An instance of a series generated with Eq. (6) is shown in Fig. 15, together with two forecasts for the next 4 values of the series. In the forecasts the multiplicative and the additive transformations have been used. Forecasts with the additive transformation underestimate multiplicative trends.

Other approaches for dealing with trending series have been developed in previous works applying GRNN to time series forecasting. In [11] a strategy similar to our additive transformation is used. However, the transformation is not applied to the training examples, but to segments in which the series are divided. In [19] the technique of taking first differences to remove trends, used in the ARIMA methodology [4], is applied successfully with multilayer perceptron. However, [11,8] also take first differences to remove trends with GRNN models and obtain poor results. Therefore, [8] recommends to apply no transformation to remove trends.

4.2. Seasonality

In our experience with GRNN, a right choice of the autoregressive lags, as explained in Section 3.3 with the heuristic strategy, together with a proper selection of the smoothing parameter is enough to deal with time series with seasonal patterns, so we do not propose any transformation or preprocessing to cope with seasonal series.

Nevertheless, previous works in time series forecasting with GRNN apply specific strategies to deal with seasonal patterns. For example, in [8] seasonal differences are taken, as in the seasonal ARIMA model [4]. Another common approach is to decompose the time series into a seasonal and a trend-remainder component. The model is trained with the trend-remainder component and forecasts are generated for this component, then these forecasts are back transformed by adding the seasonal component. There are several ways of decomposing a time series into seasonal and trend-remainder component [20]. In [11] seasonality is dealt using this kind of decomposition. In [2,21] an original approach is proposed in which a different model is used to forecast each different season; the targets of a model predicting a given season only contain values from that season.

Original examples

Training patterns	Target patterns
1, 3	6, 7
3, 6	7, 2
6, 7	2, 9
7, 2	9, 5

Transformed examples

Training patterns	Target patterns
-1, 1	4, 5
-1.5, 1.5	2.5, -2.5
-0.5, 0.5	-4.5, 2.5
2.5, -2.5	4.5, 0.5

Fig. 14. Additive transformation of the training examples.

5. Experiments

In this section we experiment with the different strategies proposed in previous sections. As data sets the 111 time series from the NN3 competition [10] and the 1428 monthly time series from the M3 competition [9] have been used. As in the original competitions, for both data sets the next 18 future values of each time series are forecast.

To assess forecast accuracy the symmetric mean absolute percentage error—sMAPE—has been chosen. The reason for this choice is that sMAPE is the main forecasting accuracy measure applied in time series competitions. Given a time series whose next h future values are predicted, sMAPE is computed as:

$$sMAPE = \frac{1}{h}\sum_{t=1}^{h} \frac{|y_t - f_t|}{\frac{|y_t|+|f_t|}{2}} 100 \tag{7}$$

where y_t and f_t are the actual future value and forecast for horizon t respectively. Given a data set, such as the time series of the NN3 competition, the sMAPE for every time series is computed using

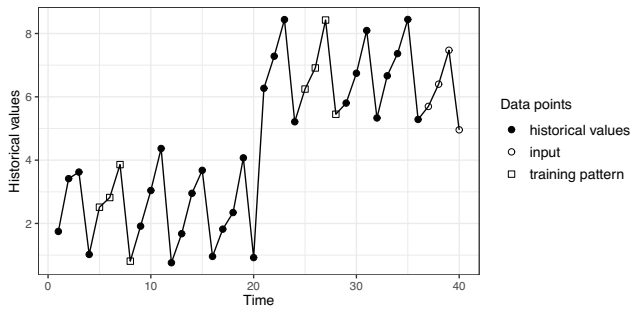

Fig. 13. Time series with a level shift.

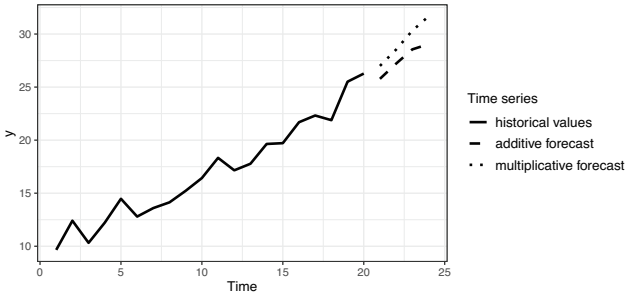

Fig. 15. Foresting a time series with multiplicative trend.

Eq. (7) and the global mean sMAPE is computed as the mean of the sMAPE across all its series. sMAPE is similar to MAPE—it represents an absolute percentage error—and it was developed because MAPE puts a heavier penalty on negative errors than on positive errors [22]. Because sMAPE is an absolute percentage error, the smaller the sMAPE value is the more accurate the forecast is.

In the next subsections the experimental results are presented.

5.1. The smoothing parameter

We have done some experimentation to see how forecast accuracy is affected by the use of rolling or fixed origin evaluation in the optimization of the smoothing parameter as proposed in Section 3.1. As optimizer the function *optim* from the *stats* package in *R* has been used. This function implements several optimization algorithms, from which the well-known Brent optimization algorithm [23] has been selected.

Also, we have implemented the straightforward strategy proposed in [8] in which the smoothing factor is selected among the fixed set of possible values [0.05, 0.1, 0.2, 0.3, 0.5, 0.6, 0.7]. From this set, the value obtaining the best accuracy on a validation set is chosen as smoothing factor.

The result of this comparison in terms of global mean sMAPE is shown in Table 1—between parenthesis the time needed, in seconds, to forecast all the data set. Rolling origin evaluation seems to choose a smoothing parameter with a better forecast accuracy, although the more complex evaluation process slows down the forecasts. The strategy proposed in [8] is very fast, but as expected it produces the worst forecasts. In this comparison the strategy for forecasting several values is MIMO—Section 3.2—and the lagged values used in the training patterns are selected with the heuristic explained in Section 3.3.

5.2. Multiple step ahead strategies

In this section an empirical comparison of the strategies for forecasting multiple future values described in Section 3.2 has been done. In this comparison the smoothing parameter is selected with rolling origin evaluation. For MIMO and the recursive strategy the previous values used as training patterns are selected with the heuristic explained in Section 3.3. For the direct strategy this heuristic cannot be applied, in this case the first 12 consecutive previous values available are used. The forecast accuracy is shown in Table 2, the recursive strategy produces the best results. However, it is slightly slower because the forecasts are generated in an iterative way.

5.3. Feature selection

In order to assess the effectiveness of the heuristic for feature selection proposed in Section 3.3, it has been compared to two classical feature selection approaches: forward selection and backward elimination [24]. These approaches use the last h values of the time series, where h is the forecasting horizon, as a validation set using rolling origin evaluation. Also, both approaches consider as possible features lagged values from 1 to 12. It must be noted that forward selection and backward elimination can select any combination of lags; specifically the lags do not have to be consecutive. Furthermore, the comparison includes the strategy applied in [11] in which the set of consecutive lags: lag 1, lags 1 to 2, lags 1 to 3, . . ., lags 1 to 12 are considered using rolling origin evaluation. That is, the combination of consecutive lags that obtains the best forecast accuracy on a validation set is chosen.

Table 3 shows a comparison of the four approaches with regard to sMAPE and computing time as previously. In this comparison the smoothing parameter has been optimized using rolling origin evaluation and the recursive approach is applied for multiple step ahead forecasts. In terms of forecast accuracy all the strategies seem to be similar. However, the proposed heuristic is much faster. Forward selection and backward elimination are remarkably slow.

Table 1
Comparison of strategies for selecting the smoothing parameter.

	NN3	monthly M3
Rolling origin	**17.7**% (33)	**18.5**% (453)
Fixed origin	18.4% (4)	19.4% (56)
Fixed set of values	20.8% (2)	25.3% (24)

Table 2
Comparison of strategies for multiple step ahead forecasts.

	NN3	monthly M3
MIMO	17.7% (33)	18.5% (453)
Recursive	**16.6**% (61)	**15.5**% (851)
Direct	20.0% (45)	17.5% (582)

5.4. Transformations: trend

In this section we experiment with the application of the additive and multiplicative transformations proposed in Section 4. In Table 4 the positive effect of these transformations on the forecast accuracy across the data sets used in previous experiments can be observed. In this comparison the lagged values have been selected using the heuristic explained in Section 3.3 and the smoothing parameter is chosen applying rolling origin evaluation. As multiple step ahead strategies MIMO and the recursive approach are used. For both data sets the forecast accuracy is improved.

In order to shed some light on whether taking first differences is useful in dealing with trending series when using GRNN we have done some experimentation. Firstly, to determine whether a time series has a trend the *ndiffs* function of the *forecast* package [25] has been used. The function *ndiffs* uses the KPSS unit root test [26] to determine the number of differences required for a time series to be made stationary. This test has considered that 61 series from the NN3 competition and 349 from the monthly series of the M3 competition have a trend. For these series we have compared three approaches: taking first differences, applying the additive transformation and applying no transformation. As strategy for forecasting multiple step ahead values the recursive approach has been used. The smoothing parameter has been optimized using the rolling origin technique and the lagged values are selected using the heuristic proposed in Section 3.3. The result of this comparison in terms of sMAPE is shown in Table 5. In line with previous works [8,11] it seems that taking first differences is not effective in dealing with trending series when GRNN is used.

5.5. Transformations: seasonality

As described in Section 4, our proposal to deal with seasonality is to select the lagged values 1 to s, where s is the length of the seasonal period, and to choose a suitable—possibly low—smoothing parameter. In this section we compare this strategy with two com-

Table 3
Comparison of feature selection approaches.

	NN3	monthly M3
Heuristic technique	16.6% (61)	15.5% (851)
Forward selection	**16.3**% (2234)	15.6% (29370)
Backward elimination	16.8% (3858)	15.6% (53474)
Best lags from: 1, 1 to 2, . . ., 1 to 12	16.6% (640)	**15.4**% (8907)

Table 4
Transformation versus no transformation.

	NN3	monthly M3
MIMO, no transformation	17.7% (33)	18.5% (453)
MIMO, multiplicative transformation	15.9% (54)	14.9% (650)
MIMO, additive transformation	**15.6**% (37)	15.5% (495)
Recursive, no transformation	16.6% (61)	15.5% (851)
Recursive, multiplicative transformation	16.1% (94)	**14.4**% (1189)
Recursive, additive transformation	15.9% (71)	**14.4**% (976)

Table 5
First differences versus other approaches.

	NN3	monthly M3
First differences	22.7%	26.7%
Additive transformation	**16.2**%	**17.8**%
No transformation	16.7%	18.4%

Table 7
Transformation strategies compared with top NN3 contenders.

ID	Method	sMAPE
B09	Wildi	14.84%
B07	Theta	14.89%
C27	Echo state networks	15.18%
B03	ForecastPro	15.44%
-	**Combination**	**15.52**%
-	**MIMO additive**	**15.61**%
-	**Recursive additive**	**15.90**%
B16	DES	15.90%
-	**MIMO multiplicative**	**15.98**%
B17	Comb S-H-D	15.93%
B05	Autobox	15.95%
-	**Recursive multiplicative**	**16.12**%
C03	Linear model + GA	16.31%
B14	SES	16.42%
B15	HES	16.49%
C46	Regression tree ensemble	16.55%
C13	*k*-NN (D'yakonov)	16.57%

mon approaches: series decomposition and seasonal differences. There are several ways of decomposing a time series in seasonal and trend-remainder components [27]. We have selected the classical multiplicative decomposition as implemented by the benchmark methods of the M4 competition [17]. The three approaches has been applied to the 58 series from the NN3 competition and the 778 monthly series from the M3 competition that have passed the seasonality test of Eq. (5). The result of this comparison in terms of forecast accuracy assessed with sMAPE is shown in Table 6. In this comparison the recursive approach has been applied to multiple step ahead forecasts and the additive transformation. The classical multiplicative decomposition obtains slightly better results than our proposal. On the other hand, the method of taken seasonal differences achieves very poor results.

To end this experimental section Table 7 compares the approaches in Table 4 that use transformations with the top contenders from the NN3 competition, which was aimed at assessing the forecast accuracy of machine learning methods. In this table, the letter B in the ID column stands for statistical benchmark and C for computational intelligence method. As can be noted, our transformation approaches are only beaten by one computational intelligence method. The result of combining these four approaches, by averaging their forecasts, is also shown in Table 7, achieving an sMAPE of 15.52%, very close to the computational intelligence winner. In this competition the sMAPE of the GRNN contender was 18.58%.

The range of the sMAPE for the GRNN model using the recursive strategy and the additive transformation when predicting the series of the NN3 data set varies between 1.8%—see Fig. 16—and 71.8%—Fig. 17. Clearly, the series in Fig. 17 is complex and our model is not able to capture its underlying behavior. Also, Fig. 18 shows how GRNN can accurately predict a time series with a seasonal behavior, in this case the sMAPE is 3.35%.

6. The tsfgrnn package

This section describes the `tsfgrnn` R package for time series forecasting using GRNN, which implements the different modeling and transforming approaches proposed above. To install the package from CRAN the `install.packages` command can be used at the console: `> install.packages("tsfgrnn")`

Table 6
Comparison of approaches for coping with seasonality.

	NN3	monthly M3
Classical multiplicative decomposition	**12.9**%	**10.2**%
Our proposal	13.3%	10.6%
Seasonal differences	19.2%	14.4%

As a first example of using the package see Fig. 19. The `library` function loads the namespace of the package. Next, the `grnn_forecasting` function is applied to forecast a time series. This is the most important function in the package, requiring at least two parameters: a time series and a forecasting horizon. `grnn_forecasting` builds a GRNN model using the historical values of the time series and predicts the next *h* future values of the series, where *h* is the forecasting horizon. The function returns an S3 object which information about the fitted model and the forecast. In the example, `pred$prediction` holds a time series with the forecast. A plot with the historical values and the forecast can be generated using the functions `autoplot` and `plot`—Fig. 20 shows the plot produced by `autoplot`.

In the previous call to `grnn_forecasting` only the two mandatory arguments were set. However, `grnn_forecasting` has optional arguments to control the modeling and transforming strategies. These arguments can be consulted, as any R function, with the `help` command and are listed next:

- `lags`: an integer vector with the lagged values used to create the training patterns. If unspecified, the heuristic explained in Section 3.3 is applied. It is also possible to use a character to specify forward selection or backward elimination.
- `sigma`: a numeric value for the smoothing parameter. If unspecified, the value is selected optimizing a forecast accuracy measure on an evaluation set consisting of the last *h* values of the time series. Rolling origin evaluation is used to assess the forecast accuracy on the evaluation set, but it is also possible to select a fixed origin evaluation.
- `msas`: the multiple step ahead strategy: recursive (the default) or MIMO.
- `transform`: how the training examples are transformed: additive (the default), multiplicative or no transformation.

In the code snippet shown in Fig. 21 the `grnn_forecasting` function is invoked to forecast the next two future values of an artificial time series. In this case, the `lags` parameter has been used to set the training patterns as lagged values 1 and 2. Also, the MIMO strategy for multiple step ahead forecasting is selected and no transformation for the training examples. It is possible to consult information about the fitted GRNN model. For example, in the code snippet the smoothing parameter, which is selected automatically, is printed. The function `grnn_examples` returns a

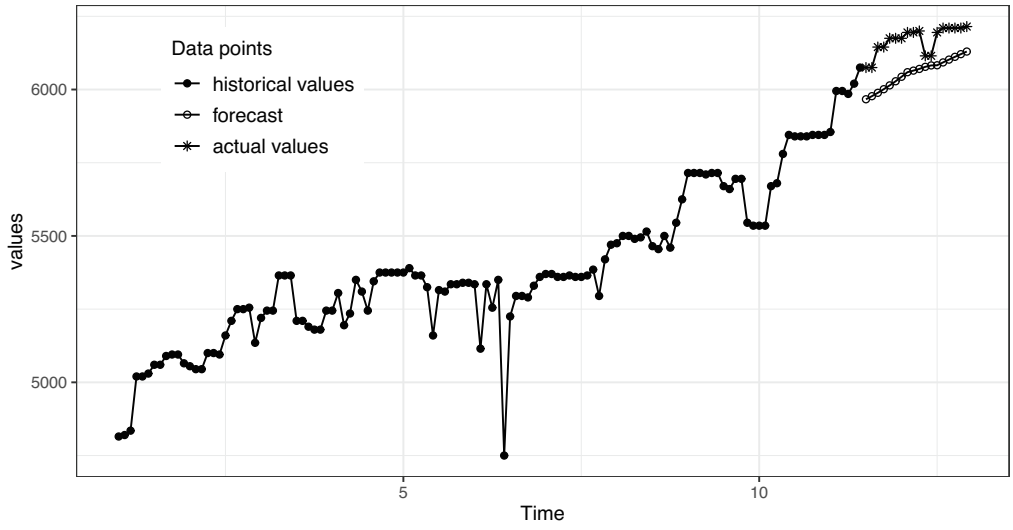

Fig. 16. Historical values and forecasts for series 77 with an sMAPE of 1.8% (best forecast).

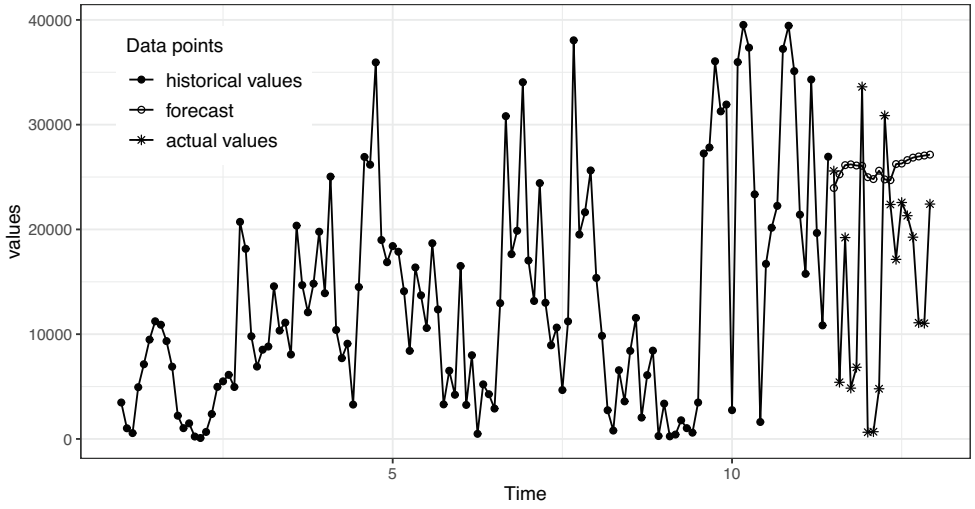

Fig. 17. Historical values and forecasts for series 93 with an sMAPE of 71.8% (worst forecast).

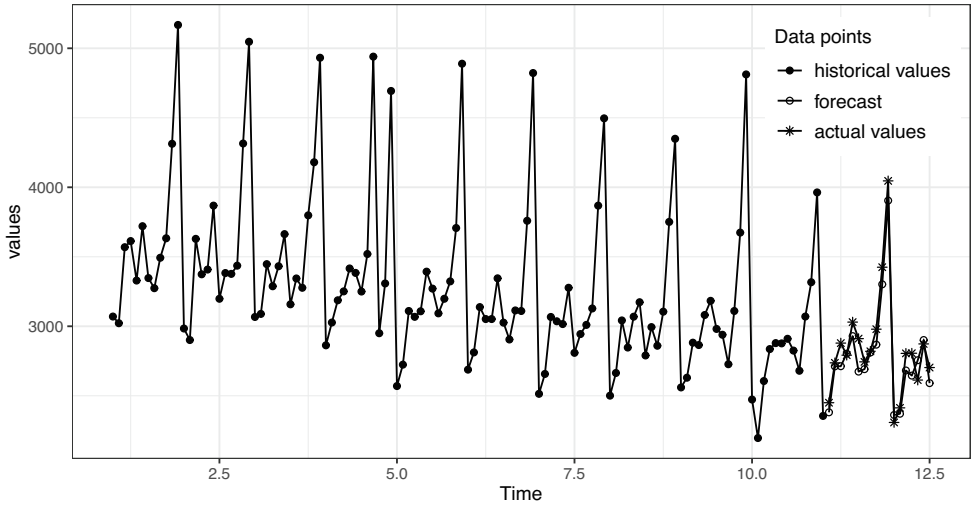

Fig. 18. Historical values and forecasts for seasonal series 68 of the NN3 competition with an sMAPE of 3.35%.

```
> library(tsfgrnn)
> pred <- grnn_forecasting(UKgas, h = 4)
> pred$prediction
         Qtr1     Qtr2     Qtr3     Qtr4
1987 1154.3363 674.6236 388.2656 868.5835
> plot(pred)
> library(ggplot2)
> autoplot(pred)
>
```

Fig. 19. Forecasting a time series with `tsfgrnn`.

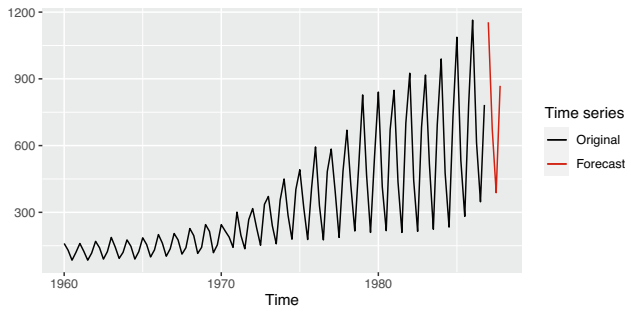

Fig. 20. Example of plot generated by `autoplot`.

data frame with the training examples. Also, the input and the weights used in the forecast can be seen with the function `grnn_weights`. In this case, the input is the vector $(11, 13)$—the last two historical values—and the only target with a significant weight is the last one. The function `plot_example` plots one of the training examples; for instance, in the snippet code the call `plot_example(pred, 1)` generates a plot showing the training example with the highest weight, see Fig. 22.

Finally, in order to assess the forecast accuracy of a GRNN model the function `rolling_origin` can be applied. Fig. 23 shows a code snippet using this function. In this code snippet, `grnn_forecasting` is used first to fit a model. Then, `rolling_origin` uses this model to assess its forecast accuracy on the

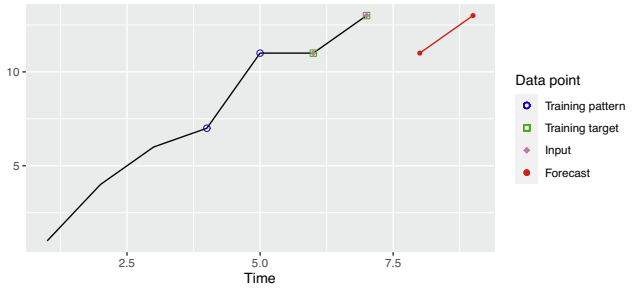

Fig. 22. `plot_example` shows a training example.

last h values of the time series using rolling origin evaluation. The code snippet shows how the different test sets, predictions and forecast errors can be consulted. Each row of the data frames represents a different test set. It is also possible to consult some forecasting accuracy measures for the whole test set or for every forecasting horizon. The generic function `plot` allows us to see a test set and its forecast. For example, the last command in the code snippet generates the plot in Fig. 24.

7. Conclusions

Generalized regression neural networks can be quickly trained, so they are very suitable for developing a fast time series forecasting tool. Nevertheless, to achieve accurate predictions several factors have to be taken into account. A proper selection of the smoothing parameter is crucial to capture different patterns of a series. In this sense, our experimentation suggests to select this parameter optimizing a forecast accuracy measure on an evaluation test formed by the last historical values of the series. To assess forecast accuracy rolling origin evaluation seems to produce better results than fixed origin evaluation, though it needs more computational resources.

Forecast accuracy is also affected by feature selection. In this work, a straightforward heuristic technique for selecting the autoregressive lagged values has been proposed. This technique helps to capture the seasonal behavior of a time series.

```
> timeS <- ts(c(1, 4, 6, 7, 11, 11, 13))
> pred <- grnn_forecasting(timeS, h = 2, lags = 1:2, msas = "MIMO", transform = "none")
> pred$model$sigma
[1] 0.692533
> grnn_examples(pred)
     Lag2 Lag1 H1 H2
[1,]    1    4  6  7
[2,]    4    6  7 11
[3,]    6    7 11 11
[4,]    7   11 11 13
> grnn_weights(pred)
$input
Lag 2 Lag 1
   11    13

$examples
     Lag2 Lag1 H1 H2       weight
[1,]    1    4  6  7 1.272749e-73
[2,]    4    6  7 11 4.833762e-36
[3,]    6    7 11 11 2.732823e-19
[4,]    7   11 11 13 1.000000e+00

> plot_example(pred, 1)
>
```

Fig. 21. Consulting the forecast of a time series.

```
Console  Terminal  R Markdown  Jobs
~/
> pred <- grnn_forecasting(ts(1:20), h = 4, lags = 1:2)
> ro <- rolling_origin(pred, h = 4)
> print(ro$test_sets)
     h=1 h=2 h=3 h=4
[1,]  17  18  19  20
[2,]  18  19  20  NA
[3,]  19  20  NA  NA
[4,]  20  NA  NA  NA
> print(ro$predictions)
          h=1      h=2      h=3      h=4
[1,] 17.10345 18.26397 19.51306 20.8425
[2,] 18.09677 19.24662 20.47863       NA
[3,] 19.09091 20.23140       NA       NA
[4,] 20.08571       NA       NA       NA
> print(ro$errors)
             h=1        h=2        h=3        h=4
[1,] -0.10344828 -0.2639715 -0.5130592 -0.8424997
[2,] -0.09677419 -0.2466181 -0.4786345         NA
[3,] -0.09090909 -0.2314050         NA         NA
[4,] -0.08571429         NA         NA         NA
> ro$global_accu
     RMSE       MAE      MAPE     SMAPE
0.3770748 0.2953034 1.5280699 1.5098488
> ro$h_accu
             h=1       h=2       h=3       h=4
RMSE  0.09444336 0.2476891 0.4961455 0.8424997
MAE   0.09421146 0.2473315 0.4958468 0.8424997
MAPE  0.51329850 1.3071743 2.5467419 4.2124983
SMAPE 0.51196211 1.2986078 2.5146067 4.1256028
> plot(ro, h = 4)
```

Fig. 23. Assessing forecast accuracy with function `rolling_origin`.

Generalized regression neural networks are not able to properly forecast a time series with a trend. In this regard, an additive and a multiplicative transformation of the training examples have been proposed. Experimental results seem to indicate that these transformations improve the forecast accuracy of GRNN.

All of the modeling and transforming strategies proposed in this paper have been incorporated into the `tsfgrnn` R package, an open source software that is available through CRAN, the official repository of R packages.

CRediT authorship contribution statement

Francisco Martínez: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Visualization. **Francisco Charte:** Conceptualization, Software, Writing - review & editing. **María Pilar Frías:** Conceptualization, Methodology, Formal analysis, Writing - review & editing. **Ana María Martínez-Rodríguez:** Methodology, Formal analysis, Writing - review & editing.

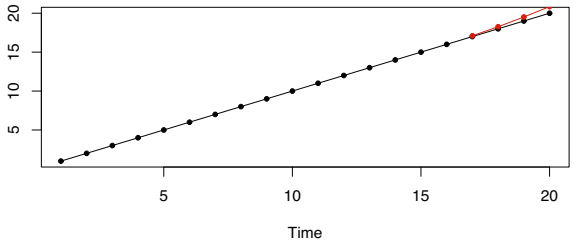

Fig. 24. A test set and its forecast.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Funding: This work was supported by the Spanish Ministry of Science, Innovation and Universities [project PID2019-107793 GB-I00]. Funding for open access charge: Universidad de Jaén / CBUA.

References

[1] K. Kim, Financial time series forecasting using support vector machines, Neurocomputing 55 (1) (2003) 307–319, https://doi.org/10.1016/S0925-2312(03)00372-2.
[2] G. Dudek, Neural networks for pattern-based short-term load forecasting: A comparative study, Neurocomputing 205 (2016) 64–74, https://doi.org/10.1016/j.neucom.2016.04.021.
[3] R. Araújo, A. Oliveira, S. Meira, On the problem of forecasting air pollutant concentration with morphological models, Neurocomputing 265 (2017) 91–104, https://doi.org/10.1016/j.neucom.2017.01.107.
[4] G.E.P. Box, G.M. Jenkins, G.C. Reinsel, Time Series Analysis: Forecasting and Control, 4th Edition., John Wiley & Sons, Hoboken, 2008.
[5] P. Montero-Manso, G. Athanasopoulos, R.J. Hyndman, T.S. Talagala, FFORMA: Feature-based forecast model averaging, International Journal of Forecasting 36 (1) (2020) 86–92, https://doi.org/10.1016/j.ijforecast.2019.02.011.
[6] D.F. Specht, A general regression neural network, Trans. Neur. Netw. 2 (6) (1991) 568–576, https://doi.org/10.1109/72.97934.
[7] D. Broomhead, D. Lowe, Multivariable functional interpolation and adaptive networks, Complex Systems 2 (1988) 321–355.

[8] N. Ahmed, A. Atiya, N.E. Gayar, H. El-Shishiny, An empirical comparison of machine learning models for time series forecasting, Econometric Reviews 29 (5–6) (2010) 594–621, https://doi.org/10.1080/07474938.2010.481556.
[9] S. Makridakis, M. Hibon, The M3-Competition: results, conclusions and implications, International Journal of Forecasting 16 (4) (2000) 451–476, https://doi.org/10.1016/S0169-2070(00)00057-1.
[10] S.F. Crone, M. Hibon, K. Nikolopoulos, Advances in forecasting with neural networks? empirical evidence from the NN3 competition on time series prediction, International Journal of Forecasting 27 (3) (2011) 635–660, https://doi.org/10.1016/j.ijforecast.2011.04.001.
[11] W. Yan, Toward automatic time-series forecasting using neural networks., IEEE Trans. Neural Netw. Learning Syst. 23 (7) (2012) 1028–1039. doi:10.1109/TNNLS.2012.2198074.
[12] F. Martínez, F. Charte, A. Rivera, M. Frías, Automatic time series forecasting with GRNN: A comparison with other models, in: International Work-Conference on Artificial Neural Networks, Springer, 2019, pp. 198–209. doi:10.1007/978-3-030-20521-8_17.
[13] L.J. Tashman, Out-of-sample tests of forecasting accuracy: an analysis and review, International Journal of Forecasting 16 (4) (2000) 437–450, https://doi.org/10.1016/S0169-2070(00)00065-0.
[14] S. Ben Taieb, G. Bontempi, A.F. Atiya, A. Sorjamaa, A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition, Expert Syst. Appl. 39 (8) (2012) 7067–7083. doi:10.1016/j.eswa.2012.01.039.
[15] E.S. Gardner, Exponential smoothing: The state of the art – part ii, International Journal of Forecasting 22 (4) (2006) 637–666, https://doi.org/10.1016/j.ijforecast.2006.03.005.
[16] F. Martínez, M.P. Frías, M.D. Pérez, A.J. Rivera, A methodology for applying k-nearest neighbor to time series forecasting, Artificial Intelligence Review 52 (3) (2019) 2019–2037, https://doi.org/10.1007/s10462-017-9593-z.
[17] S. Makridakis, E. Spiliotis, V. Assimakopoulos, The M4 Competition: 100,000 time series and 61 forecasting methods, International Journal of Forecasting 36 (1) (2020) 54–74, https://doi.org/10.1016/j.ijforecast.2019.04.014.
[18] Y. Kang, E. Spiliotis, F. Petropoulos, N. Athiniotis, F. Li, V. Assimakopoulos, Déjà vu: A data-centric forecasting approach through time series cross-similarity, Journal of Business Research doi:10.1016/j.jbusres.2020.10.051.
[19] M. Qi, G.P. Zhang, Trend time–series modeling and forecasting with neural networks, IEEE Transactions on Neural Networks 19 (5) (2008) 808–816, https://doi.org/10.1109/TNN.2007.912308.
[20] R.B. Cleveland, W.S. Cleveland, J.E. McRae, I. Terpenning, STL: A Seasonal-Trend Decomposition Procedure Based on Loess, Journal of Official Statistics 6 (1) (1990) 3–73.
[21] F. Martínez, M.P. Frías, M.D. Pérez-Godoy, A.J. Rivera, Dealing with seasonality by narrowing the training set in time series forecasting with kNN, Expert Systems with Applications 103 (2018) 38–48, https://doi.org/10.1016/j.eswa.2018.03.005.
[22] R.J. Hyndman, A.B. Koehler, Another look at measures of forecast accuracy, International Journal of Forecasting 22 (4) (2006) 679–688, https://doi.org/10.1016/j.ijforecast.2006.03.001.
[23] R. Brent, Algorithms for minimization without derivatives, Prentice-Hall, 1973.
[24] A.A. Freitas, Data Mining and Knowledge Discovery with Evolutionary Algorithms, Springer-Verlag, New York Inc, Secaucus, NJ, USA, 2002.
[25] R.J. Hyndman, Y. Khandakar, Automatic Time Series Forecasting: The forecast Package for R, Journal of Statistical Software 27 (3) (2008) 1–22, https://doi.org/10.18637/jss.v027.i03.
[26] D. Kwiatkowski, P.C. Phillips, P. Schmidt, Y. Shin, Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?, Journal of Econometrics 54 (1) (1992) 159–178, https://doiorg/10.1016/0304-4076(92)90104-Y.
[27] J. Ord, R. Fildes, N. Kourentzes, Principles of Business Forecasting–2nd Ed, wessex, Incorporated, 2017.

Francisco Martínez. Assistant professor with the Computer Science Dept. of the University of Jaén, Spain. He received his B.S. degree in Computer Science from the University of Granada and his Ph.D. from the University of Jaén. He has worked in the fields of parallel programming and computer graphics. Currently, his main research topic is time series forecasting.

Francisco Charte. T. Eng. Computer Science (2008) and B. Eng. Computer Science (2010) from the Universidad de Jaén, with Extraordinary Award in both degrees and 1st National Award for Excellence in Academic Performance from the MECD (2010). M.Sc. Soft Computing and Computational Intelligence (2011) and PhD from the Universidad de Granada (2015).
He is currently an Assistant Professor with the Computer Science Dept., Universidad de Jaén, Spain. He is the author of more than 130 books, including the title "Multilabel Classification. Problem analysis, metrics and techniques" published by Springer, as well as author of 25 JCR research papers and several dozen contributions to international conferences. His main research interests include multilabel learning, imbalanced and high-dimensionality problems and representation learning through deep learning techniques.

María P. Frías. Assistant professor with the Statistics and Operation Research Dept. at the University of Jaén (Spain). She received her B.S. degree in Statistical Sciences and Techniques from the University of Granada and her Ph.D. from the University of Jaén. Her research interests include topics like spatial statistics, long-range dependence random field models and parameter estimation of spatiotemporal random field models.

Ana María Martínez-Rodríguez. Assistant professor in the Department of Statistics and Operational Research at University of Jaén (Spain). She has a PhD in Mathematics (University of Jaén, Spain). Her current research interests include Regression Models for Count Data, Probabilistic Distributions and Applied Statistics.