XML-IS

Our reference: NEUCOM 12341 P-authorquery-vx

AUTHOR QUERY FORM

ELSEVIER	**Journal: NEUCOM** **Article Number: 12341**	**Please e-mail or fax your responses and any corrections to:** **E-mail: corrections.esch@elsevier.macipd.com** **Fax: +44 1392 285878**

Dear Author,

Please check your proof carefully and mark all corrections at the appropriate place in the proof (e.g., by using on-screen annotation in the PDF file) or compile them in a separate list.

For correction or revision of any artwork, please consult http://www.elsevier.com/artworkinstructions.

Any queries or remarks that have arisen during the processing of your manuscript are listed below and highlighted by flags in the proof. Click on the Q link to go to the location in the proof.

Location in article	Query / Remark: click on the Q link to go Please insert your reply or correction at the corresponding line in the proof
Q1	Please confirm that given names and surnames have been identified correctly.
Q2	The citation of "Section [2.3]" has been modified to "Section [2.2.3]". Please check, and correct if necessary.
Q3	To maintain sequential order, Eq. [6] has been changed to Eq. [5]. Please check, and correct if necessary.
Q4	The given sentence seems to be incomplete. Please check for missing words/phrases and complete the sentence.
Q5	Fig. 5 has not been provided but is cited in the text. Please supply the figure or delete the citation from the text.
Q6	To maintain sequential order, Eq. [B3] has been changed to Eq. [B2]. Please check, and correct if necessary.
Q7	Please check the page range in reference: [14].

Thank you for your assistance.

Neurocomputing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Neurocomputing

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

A novel neural network ensemble architecture for time series forecasting

Q1 Iffat A. Gheyas*, Leslie S. Smith

University of Aberdeen Business School, Edward Wright Building, Dunbar Street, Old Aberdeen AB24 3QY, United Kingdom

ARTICLE INFO

Article history:
Received 3 September 2010
Received in revised form
7 June 2011
Accepted 3 August 2011
Communicated by G.P. Zhang

Keywords:
Time series forecasting
Generalized regression neural networks
Neural network ensemble
Curse of dimensionality
Deseasonalization
Dynamic nonlinear weighted voting

ABSTRACT

We propose a novel homogeneous neural network ensemble approach called Generalized Regression Neural Network (GEFTS–GRNN) Ensemble for Forecasting Time Series, which is a concatenation of existing machine learning algorithms. GEFTS uses a dynamic nonlinear weighting system wherein the outputs from several base-level GRNNs are combined using a combiner GRNN to produce the final output. We compare GEFTS with the 11 most used algorithms on 30 real datasets. The proposed algorithm appears to be more powerful than existing ones. Unlike conventional algorithms, GEFTS is effective in forecasting time series with seasonal patterns.

1. Introduction

Time-series forecasting is about projecting data into the future. Univariate time-series datasets contain values for a single variable across multiple time periods. Time series violate at least two fundamental assumptions of spatial statistics: independence and identity of distribution (i.i.d.). These limitations pose significant challenges for computational modelling of data. Severe multicollinearity is a direct consequence of the departure from the independence assumption. In time series analysis, the input variables are the values of the given variable at selected lags and the output variables are the forecasts for different horizons. Lag variables are autocorrelated, and hence it is difficult to estimate individual response coefficients accurately. One way to reduce the impact of collinearity is to increase sample size. Since time series data follow one variable's changes over time, lack of training data is a burning issue in time series applications. Shortage of degrees of freedom and severe **multicollinearity in time-series** data often frustrate data miners. To make matters worse, real world time series often violate the assumption of identity (stationarity) so that the probability laws (i.e. the multivariate joint distribution) governing the process change with time or space—that is, the process is not in statistical equilibrium. This makes time series data inherently high dimensional and not reducible to two or three dimensions. A variety of modelling techniques are available for this problem including Autoregressive Integrated Moving Average (ARIMA) and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) [1], Artificial Neural Networks (ANNs) [2] and Support Vector Machines (SVMs) [3]. These conventional algorithms often have many drawbacks, such as local optima, overfitting, dimension disasters, etc., which make the algorithms inefficient.

We present a novel algorithm for forecasting univariate time series and future volatility. Our algorithm is a concatenation of well respected algorithms. The remainder of this paper is organized as follows: the proposed algorithm in Section 2, research methodology in Section 3, results and discussion in Section 4, followed by summary and conclusions in Section 5.

2. Proposed algorithm (GEFTS)

We propose a new efficient time series forecasting algorithm, Generalized Regression Neural Network (GEFTS–GRNN [4]) Ensemble for Forecasting Time Series, which combines several well respected algorithms. GEFTS consists of two layers. In the

Abbreviations: ACF, Autocorrelation function; ANN, Artificial Neural Networks; ARIMA, Autoregressive Integrated Moving Average; BIC, Bayesian Information Criterion; d, deseasonalized data; ERNN, Elman's Recurrent Neural Networks; GA, Genetic Algorithm; GARCH, Generalized Autoregressive Heteroskedasticity; GRNN, Generalized Regression Neural Networks; HA, Hybrid Algorithm of regression-based methods and ERNN; HC, Hill-Climbing; HES, Heterogeneous Ensemble with Simple averaging approach; HEW, Homogeneous ensemble with weighted averaging approach; HOS, Homogeneous ensemble with Simple averaging approach; HOW, Homogeneous ensemble with static Weighted averaging approach; HUX, Half Uniform Crossover; JE, Jacobs' Ensemble; MLP, Multilayer Perceptrons; nd, non-deseasonalized data; NN, Neural Networks; PACF, Partial Autocorrelation Function; PSO, Particle Swarm Optimization; RBFN, Radial Basis Function Networks; SA, Simulated Annealing; SVM, Support Vector Machines.

* Corresponding author. Tel.: +44 01224 272212, Mob.: 07776001754; fax: +44 01224 272181.
E-mail address: iag@cs.stir.ac.uk (I.A. Gheyas).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2011.08.005

first layer, it has a set of individual GRNN learners (called base learners) and in the second has a single GRNN (called the combiner). The overall architecture of GEFTS is presented in Fig. 1. Each base learner is trained on a different subset of features. No feature can be present in two different subsets. The outputs of selected networks are then used to train a new GRNN to combine the predictions of individual networks. GEFTS applies SAGA [5] to optimize feature subsets and the ensemble of multiple base learners.

The advantages of GEFTS are summarized in Section 2.1. In Section 2.2, we discuss the implementation process of GEFTS. Q2 Section 2.2.3 provides the justifications behind the decisions taken during the selection of underlying algorithms and their associated parameters in GEFTS.

2.1. Advantages of proposed algorithm (GEFTS) over conventional forecasting techniques

(I) The primary advantage of GEFTS is that it is robust against the curse of dimensionality. This is because GEFTS is a mixture of GRNNs where each GRNN is trained on a small subset of the whole feature space—this ensemble architecture enables GEFTS to maintain an adequate ratio of training cases per variable to reduce the impact of high dimensional datasets. Detrending and deseasonalization are critical for conventional time series forecasting algorithms. If there is a long-term trend in the data, the joint probability laws will be different at every time step. Under such circumstances, no machine learning algorithm can learn. In contrast, the characteristics of joint distributions change cyclically across the seasonal time series. Theoretically, seasonal patterns should be predictable. The model of both detrended and deseasonalized data only includes non-seasonal lags, whereas the models of time series with seasonal patterns must include both seasonal and non-seasonal lags that give rise to an increase of the effective dimensionality (here, "effective dimensionality" refers to the smallest number of predictor variables that are necessary to model the time series adequately). In time series, the available sample size shrinks dramatically when the model needs to include long seasonal lags. This leads to a very inefficient model and the forecasting becomes less reliable. Therefore, applying both detrending and deseasonalization simultaneously is an effective data pre-processing approach in modelling and forecasting time series with conventional forecasting techniques. However, there are no hard and fast rules for the identification of the seasonal patterns in the data. Hence, it can be very difficult to identify periodicities from random patterns (noise), which often leads to underdifferencing or overdifferencing. Eggleton showed that subjects were unable to distinguish between alternating sequences and random sequences in contrived time series data [5]. The underdifferenced series exhibits seasonal variations which results in poor forecasting performance due to the curse of dimensionality. The effects of overdifferencing are even worse. Overdifferencing may lead to an efficiency loss, and a possible deterioration of forecasting, since overdifferencing can introduce patterns into the original observations which were not actually in the data before the differencing.GEFTS is able to model seasonality directly so that prior deseasonalization is not necessary. Because GEFTS is robust against the curse of dimensionality, it can take a large number of lagged variables into account without overfitting. Hence, GEFTS can deal with seasonal time series problems.

(II) GEFTS uses a dynamic nonlinear weighted voting scheme to combine the base classifiers into a single, composite classifier. Each base GRNN learner independently forecasts the output. A combiner GRNN is then trained to predict the final output from the outputs of base learners. In essence, the pattern of base learners' outputs provides the combiner learner indirect feedback about the correctness of its base learners for a given input. Consequently, GEFTS can capture interactions among base learners.

2.2. Design and implementation strategy of GEFTS

GEFTS forecasts both future values of a time series and expected future volatility. For this, GEFTS constructs two ensemble models—model-P and model-Q. Model-P forecasts expected future values, whereas model-Q forecasts expected future volatilities. Model-P is fitted to the time series data and model-Q is fitted to the square of residuals obtained from the model-P on the training dataset. Model-P is developed before model-Q.

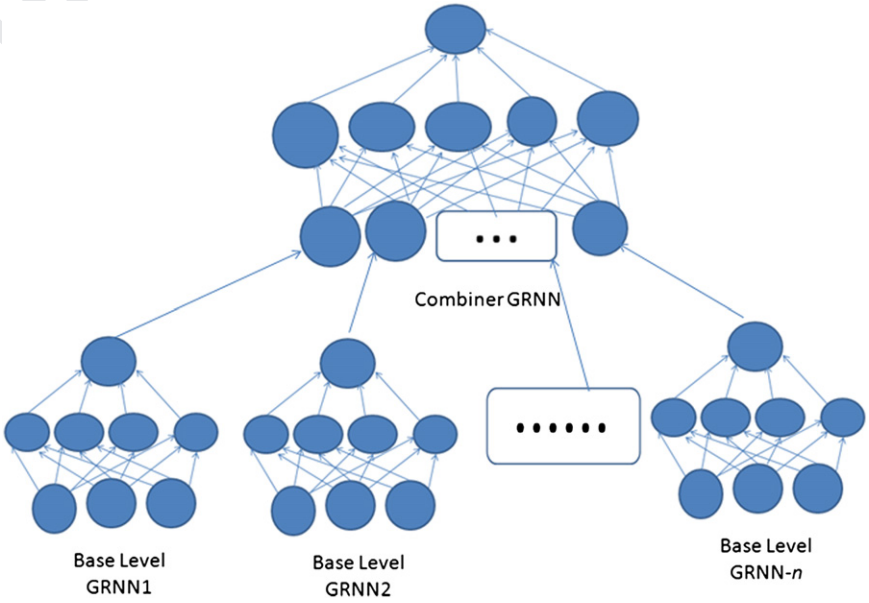

Fig. 1. Architecture of GEFTS.

Please cite this article as: I.A. Gheyas, L.S. Smith, A novel neural network ensemble architecture for time series forecasting, Neurocomputing (2011), doi:10.1016/j.neucom.2011.08.005

Once the models are constructed, they forecast future events independently based on historical information. The pseudo code of GEFTS is presented in Section 2.2.1. GEFTS employs SAGA [6] to optimize the ensemble makeup. Section 2.2.2 explains how SAGA evaluates alternative feature subsets for base learners and the combiner learner.

Appendix A presents a brief overview of GRNN and Appendix B provides a short overview of SAGA.

2.2.1. Pseudo code of GEFTS

GEFTS constructs an ensemble model for 1-step-ahead forecasts. GEFTS performs multi-step-ahead forecasting by iteratively making repeated one-step-ahead predictions up to the desired horizons feeding predictions back in as inputs in order to forecast further into the future.

Step 1: Construction of the model (P) for forecasting future values in the time series:

Step 1.1: Detrend the time series. Section 3.2 describes how we detrend the time series.

Step 1.2: Select time lags with no less than 10 observations as candidate input variables. The outputs are one-step-ahead time series values.

Step 1.3: The purpose of this step is to partition the feature space into a finite set of optimal mutually exclusive and exhaustive subspaces. GEFTS does this in the following way:

(I) Apply SAGA to the chosen feature space to select the best feature subset.
(II) Remove features included in the selected feature subset from the original feature space before selecting the next best feature subset solution.

Repeat steps I to II until there are no features left in the feature space.

Step 1.4: Train a GRNN with each feature subset selected in step 1.3. Each GRNN is trained to forecast the 1-step-ahead value. During training, we set the default value for each centre's width σ to 0.5 times the average distance to 20 nearest neighbours (see Section 2.2.3.2).

Step 1.5: Present the training patterns one after the other to each of the trained GRNNs for the prediction of output values.

Step 1.6: Generate a training dataset where the input vectors consist of the forecasts of the GRNN learners. That is, the generated dataset consists of a number of different features where each feature denotes the forecasts by one trained GRNN for the original training patterns. Hence, the number of input variables in the new dataset is equal to the number of trained GRNNs. The output variable is actual values of one-step-ahead.

Step 1.7: Run SAGA on the new dataset to select an optimal subset of features for forecasting the k-step-ahead value.

Step 1.8: Select the GRNNs, from which the selected features for the combiner GRNN were derived, as base classifiers.

Step 1.9: Finally, train a combiner GRNN on the new dataset (i.e. predictions of base level GRNNs) using the selected feature subsets.

Step 2: Construction of GRNN ensemble model 'Q' for the forecasting volatility of time-series:

Step 2.1: Present training patterns to the model 'P' for prediction purposes. Then find the squared residual series a_t by subtracting the predicted value $\hat{Z}_t$ from the actual value Z_t and by taking the square of the residual:

$$a_t = (Z_t - \hat{Z}_t)^2. \quad (1)$$

Step 2.2: We fit a GRNN ensemble model to this squared residual series as we did for the original time series data in step 1. We name this model as 'Q'. Model 'Q' forecasts future volatility (the conditional mean of the squared residuals) based on the lagged squared residuals. It is very important to note that during the out-of-sample forecasting, the process variance increases linearly with time due to the accumulation of random noise. Hence GEFTS calculates conditional variance at t-steps-ahead (where $h=1, 2, 3,\ldots$) out-of-sample forecasting, using Eq. (2).

Variance at time horizon t:

$$\mathrm{Var}(\hat{Z}_t) = (\text{Predicted squared residual at time step } t) \times t \quad (2)$$

We compute a 95% confidence interval (C.I.) for the forecast of the future value of time series ($\hat{Z}_t$) as in Eq. (3)

$$95\%\ \text{C.I.} = \hat{Z}_t \pm 1.96\mathrm{Var}(\hat{Z}_t) \quad (3)$$

Step 3: The forecasts of models P and Q must finally be numerically inverted **back to the original time domain**.

2.2.2. How SAGA assigns fitness scores to candidate solutions

The use of the validation set for feature subset selection would reduce the amount of data available for training significantly. Hence, we evaluate alternative feature subsets on the full set of training data using Bayesian Information Criterion (BIC).

The fitness value is assigned to each feature subset solution according to the BIC value. The BIC value was used to evaluate the feature subset solution. The lower the BIC values, the higher the fitness scores.

$$BIC = n\left[\ln\left(\frac{RSS}{n}\right)\right] + (\ln(n))k \quad (4)$$

where RSS=residual sum-of-squares on the training data from the estimated model, n=number of data points in training set and k=number of features in a feature subset.

SAGA assigns rank 1,…,m to the solutions according to their BIC values, where rank '1' is the worst-fitness solution. For each solution, we divide its rank value by the rank sum (i.e. the sum of the ranks of the solutions evaluated so far) to obtain the fitness score. The fitness scores of all the solutions were updated following the evaluation of each new solution.

2.2.3. Justification of design choices and parameter settings chosen for SAGA

2.2.3.1. Reasons behind using GRNN as a machine learning algorithm. GRNN has many advantages including its non-parametric nature, local approximation property and only one adjustable parameter. Moreover, it **is a computationally very efficient** approach since it is a one-pass algorithm.

2.2.3.2. Reasons for setting the smoothing parameter of GRNN to 0.5 times of the average distance to 20 nearest neighbours. GRNN has only one adjustable parameter (the smoothing parameter σ). However, we have discovered the best default value for σ. We set the default value for each centre's width σ to 0.5 times of the average distance to 20 nearest neighbours. The default parameter setting was adopted on 200 synthetic datasets with various data characteristics (Appendix C).

With this parameter setting, the width of each prototype pattern (i.e. centre) within a GRNN is different since each prototype pattern has a unique set of 20 different neighbours. As a result, the proposed default value of σ can keep the local approximation ability of GRNNs intact. Appendix C shows performance impact of different σ values.

2.2.3.3. Reasons behind using SAGA as a search algorithm. None of the popular optimization algorithms are perfect in the sense that no

Please cite this article as: I.A. Gheyas, L.S. Smith, A novel neural network ensemble architecture for time series forecasting, Neurocomputing (2011), doi:10.1016/j.neucom.2011.08.005

single technique fully satisfies all the requirements of an optimal search algorithm. Simulated Annealing (SA) suffers from slow convergence. Genetic Algorithm (GA) suffers from premature convergence. Hill-climbing (HC) always gets stuck into the nearest local minimum to the start position. However, the hybrid of these three algorithms can overcome the weaknesses of each. SAGA is a hybrid scheme that sequentially applies: (i) a SA, (ii) a GA and (iii) a HC. SAGA may not always find the best solution but is guaranteed to find a good solution at any given time: guaranteeing the best solution is only possible using brute force search. For the comparative assessment of the search algorithms in terms of accuracy, convergence time and parsimony of the solution, interested readers are referred to [6].

3. Comparative performance analysis

We compare GEFTS with popular time series forecasting algorithms: (i) ARIMA–GARCH [1], (ii) Generalized Regression Neural Networks (GRNN) [7], (iii) feedforward Multilayer Perceptrons (MLP) [8], (iv) Elman's Recurrent Neural Networks (ERNN) [9], (v) a hybrid algorithm of regression-based methods and ERNN (HA) [10], (vi) Radial Basis Function Neural Networks (RBFNNs) [11], (vi) a homogeneous ensemble of GRNNs with simple averaging approach (HOS) [12], (vii) a homogeneous ensemble of GRNNs with static weighted voting approach (HOW) [13], (viii) a heterogeneous ensemble of a MLP, a ERNN, a GRNN, a RBFNN and a SVM) with simple averaging approach (HES) [14], (ix) a heterogeneous ensemble with static weighted voting approach (HEW) [15], (x) Jacob's dynamic ensemble approach (JE—in JE, we used GRNNs as base learners and the gating network) [16] and (XI) Support Vector Machines (SVMs) [3]. We implement these algorithms using the toolboxes MATLAB and Peltarion Synapse Neural Network Software. Interested readers should refer to the references mentioned above for the details about these algorithms. We tested forecasting algorithms on 30 real world datasets (described in Section 3.1).

In this study, we adopted the following strategies to assess the performance of proposed and existing algorithms:

- **Detrending** and deseasonalization were used as a **pre-processing** step to prepare **time series** for analysis (exactly how we do this is discussed in Section 3.2).
- For the simplicity, all forecasting algorithms (except for ARIMA–GARCH) include only autoregressive terms (lagged values of the response variable) in their models. ARIMA–GARCH uses both autoregressive terms and moving average terms (lagged errors) for prediction.
- All forecasting algorithms use SAGA for optimal feature (autoregressive and moving average terms) subset selection.
- GEFTS, Jacob' Ensemble (JE), HOS and HOW employ SAGA to select an optimal set of base learners.
- Conventional forecasting algorithms used PSO (Particle Swarm Optimization) to optimize model parameters (e.g. the values of cost parameters in SVM, the number of hidden units in NN, the vote weights of base classifiers in HOW, HEW, etc.).
- In ARIMA–GARCH, the fitted ARIMA model forecasts future values of a time series and the GARCH model forecasts future volatilities. Neural Networks (all ensemble and neural network classifiers) learn two models, one for the future values of the time series and one for future volatilities. First, a model is fitted to the actual time series data to forecast the future value, and the in-sample squared residuals of the model are obtained. Next, a second model is fitted to the squared residuals to forecast future volatilities on time series data.
- We split the time series into training and test sets. Eleven consecutive observations after every 22 consecutive steps were used as test cases. The remaining observations were used as training samples.
- All time series forecasting algorithms use the BIC to choose among tentative models with different parameter values.
- The one-step-ahead process is iterated to obtain multi-step-ahead forecasts.
- All algorithms were run on a 3.40 GHz Intel Pentiums D CPU with 2 GB RAM.
- We compare the performance of the forecasting algorithms in terms of the accuracy of point and interval forecasts on the test case domain.
- An interval forecast is considered to be correct if the actual value falls inside the predicted 95% confidence interval. Point estimation accuracy was measured using the Mean Absolute Percentage Error (MAPE) of forecasts:

$$\text{Point estimation accuracy} = 100 - \frac{100}{N}\sum_{i=1}^{N}\frac{\left|(Y_i - \hat{Y}_i)\right|}{Y_i} \quad (5)$$

where N=number of observation in the test set, Y_i=actual output and $\hat{Y}_i$=forecasted output.

- The performance of each forecasting algorithm was evaluated on the detrended time series before and after deseasonalizing data.
- We compute the accuracy of algorithms on 1, 5 and 10-step-ahead forecasts
- In time series forecasting, the magnitude of the forecasting error increases over time, since the uncertainty increases with the horizon of the forecast. When forecasting time series, interval estimates are more informative than simple point estimates. Hence, for each dataset, the algorithms were ranked in terms of their accuracy in the interval estimation. If two algorithms have the same interval-estimation accuracy on a dataset, the algorithms were ranked based on the point estimation accuracy. The Friedman test was used to test the null hypothesis that the performance is the same for all algorithms. After applying the Friedman test and noting it is significant "Comparison of Groups or Conditions with a Control" tests (details are available in [17], p. 181) were performed in order to test the (null) hypothesis that there is no significant difference between any pair of algorithms.

3.1. Description of datasets used for comparison of forecasting algorithms

We compare the proposed and conventional forecasting approaches on 30 real-world datasets, which are all from [18]. These datasets contain no missing values. In our study, we used following datasets: **(1) *The Bank Loan rate (daily)***: contains the daily observations for the period August 4, 1954–February 14, 2010. **(2) *Discount Window Primary Credit (daily)***: contains the daily observations for the period January 9, 2002–February 14, 2010. **(3) *Federal Funds***: contains the daily observations for the period July 1, 1953–February 14, 2010. **(4) *3-month Treasury Bills-Secondary Market (daily)***: contains the daily observations for the period January 4, 1953–February 14, 2010. **(5) *3-month Bankers Acceptances (top rated) (daily)***: contains the daily observations for the period April 2, 1964–June 30, 2010. **(6) *5-year inflation (indexed): TIPS (daily)***: contains the daily observations for the period November 2, 2002–December 31, 2010. **(7) *Yield on US Treasury bonds with maturity over 10 years***: contains the daily observations for the period July 1, 1981–June 30, 2000. **(8) *1-year Interest Rate Swap***: contains the daily observations for the period May 3, 2000–December 31, 2010. **(9) *Moody's Seasoned Aaa (daily)***: contains the daily observations for the period January 3, 1982–December 31, 2010. **(10) *Moody's Seasoned Baa (daily)***: contains the daily observations for the

Please cite this article as: I.A. Gheyas, L.S. Smith, A novel neural network ensemble architecture for time series forecasting, Neurocomputing (2011), doi:10.1016/j.neucom.2011.08.005

period January 3, 1982–December 31, 2010. **(11)** ***A Rating-Utility Bonds***: contains monthly values from January, 1970, through September, 1998. **(12)** ***Real Gross National Product***: contains monthly values from January, 1969, through April, 2009. **(13)** ***Real Imports***: contains monthly values from January, 1969, through April, 2009. **(14)** ***Real Exports:*** contains monthly values from January, 1969, through April, 2009. **(15)** ***Manufacturers' New Orders Durable Goods***: contains monthly values from January, 1992, through February, 2010. **(16)** ***Housing Starts***: contains monthly values from January, 1969, through February, 2010. **(17)** ***Real Fixed Residential Investment***: contains monthly values from January, 1969, through April, 2009. **(18)** ***Real Fixed non-residential Investment***: contains monthly values from January, 1969, through April, 2009. **(19)** ***Civilian Labour Force***: contains monthly values from January, 1969, through March, 2010. **(20)** ***Unemployment Rate***: contains monthly values from January, 1969, through March, 2010. **(21)** ***Manufacturers' New Orders***: contains monthly values from January, 1992, through February, 2010. **(22)** ***U.S. Employed: Household Survey***: contains monthly values from January, 1969, through March, 2010. **(23)** ***Capacity Utilization—Manufacturing***: contains monthly values from January, 1969, through February, 2010. **(24)** ***Consumer Price Index***: contains monthly values from January, 1969, through February, 2010. **(25)** ***US Employment Cost Index: Private Industry***: contains monthly values from April, 1979, through April, 2009. **(26)** ***U.S. Nominal Retail Sales***: contains monthly values from January, 1992, through February, 2010. **(27)** ***U.S. Payroll Employment—Mining***: contains monthly values from January, 1969, through September, 2009. **(28)** ***U.S. Payroll Employment—Total Non-Agricultural***: **contains monthly values from January, 1969, through March, 2010. (29)** ***11th District dry land Value***: contains monthly values from March, 1976, through March, 2010. **(30)** ***11th District crop land Value***: contains monthly values from March, 1976, through April, 2010.

Characteristics of two interesting time series are discussed in Appendix D, which may be found at http://www.cs.stir.ac.uk/~lss/research/GheyasSmith2011/GheyasSmithAppendixD_June2011.pdf.

3.2. Description of data pre-processing

In this section, we explain how we detrend and deseasonalize the data.

3.2.1. Description of detrending methods

(i) *Eliminate non-stationarity in variance:* Ascertain if there is any evidence of non-stationarity in variance by examining time series plots and by the *F*-test of variances of the first and second halves of the time series with a significance level (α) of 0.05. If the variance is not stable through the time, transform the original series into logarithms or square roots to stabilize its variance. If the logarithmic transformation does not stabilize the series variance, then try transforming the original series into its square roots.

(ii) *Eliminate non-stationarity in mean:* Examine the series for stability of its mean by observing time series plots. This can be confirmed by a *t*-test of the means of the first half of the period and the last half with a significance level (α) of 0.05. If the mean is not stable (i.e. the series indicates a trend), resolve this through first or second order non-seasonal differencing of the series.

Non-seasonal Differencing is a popular and effective way of removing trend from a time series. The strategy is to apply successive differencing until the differenced series become stationary. If the series has positive autocorrelations out to a high number of lags (Appendix Fig. E3), then it probably needs a higher order of differencing. The correlation of a variable with itself over successive time intervals is called autocorrelation (ACF) or serial correlation. First order differencing is usually sufficient to obtain trend stationarity. However, higher-order differencing was applied when necessary. The new series $\{X_1,X_2,\ldots,X_{N-1}\}$ is formed from the original series $\{Z_1,Z_2,\ldots,Z_N\}$ by first-order differencing:

$$Z_t = Z_{t+1} - Z_t = \nabla X_{t+1}$$

Occasionally second-order differencing is required using the operator ∇^2, where

$$\nabla^2 X_{t+2} = \nabla X_{t+2} - \nabla X_{t+1} = X_{t+2} - 2X_{t+1} + X_t$$

Differencing tends to introduce negative correlation. If the series initially shows strong positive autocorrelation, then a non-seasonal difference will reduce the autocorrelation. A detrended time series looks flat.

3.2.2. Description of deseasonalization methods

If there are recurring patterns in the trend stationary time series, some form of ***seasonal differencing*** was considered to make the data stationary. The autocorrelation of a periodic function is itself, periodic with the very same period. We plot the detrended data (flat time series) and examine the autocorrelation function of the data to see if there exist significant autocorrelations at the seasonal lags s, where $s>1$ (for example, $s=12$, 24, 36 and 48).

If there appear fairly significant autocorrelations at the seasonal lags, we need to assume that seasonality is playing a significant role in determining the variation in this data. In general, when we have "flat" time series data we can simply plot the sample ACF of the data and see if there are "spikes" in it and possibly around the seasonal lags of s, $2s$, $3s,\ldots$, $4s$, etc. If there are, then more likely the data has seasonality in it and some form of seasonal differencing was considered to make the data stationary. We perform the first order seasonal span differencing operation as follows: $\nabla_s X_t = X_t - X_{t-s}$.

If the lag-1 autocorrelation is zero or even negative, or the autocorrelations are all small and pattern-less, then the series does not need further differencing. The common wisdom is that "over-differencing" should be avoided since over-differencing can introduce patterns into the original observations which were not actually in the data before the differencing. If the lag-1 autocorrelation is more negative than -0.5 (and theoretically negative lag-1 autocorrelation should never be greater than 0.5 in magnitude) this may mean the series has been over-differenced.

Another symptom of possible over-differencing is an increase in the standard deviation, rather than a reduction, when the order of differencing is increased. The optimal order of differencing is often the order of differencing at which the standard deviation is lowest. The time series plot of an over-differenced series may look quite random at first glance, but if we look closer we will see a pattern of excessive changes in sign.

4. Results and discussion

We compare the performance of our proposed algorithm (GEFTS) with several well-known forecasting algorithms using 30 time-series datasets, both before and after seasonal adjustments are made (***here,'d' represents the deseasonalized data and 'nd' the non-deseasonalized data)***. The one, five and ten-step-ahead predictions are studied. We evaluated algorithms in terms of accuracy of both point and interval predictions, and RMSE.

Please cite this article as: I.A. Gheyas, L.S. Smith, A novel neural network ensemble architecture for time series forecasting, Neurocomputing (2011), doi:10.1016/j.neucom.2011.08.005

Figs. 2 and 3 compare accuracies of interval and point forecasts produced by top five forecasting algorithms.

The overall point forecast results (based on both accuracy and RMSE) of all algorithms are reported in detail in Appendix Table E1. In addition, the point forecast accuracy and RMSE of GEFTS both before and after deseasonalizing for each of the 30 datasets are provided in

Friedman Test reveals significant differences ($p < 0.05$) in the performance of time series forecasting algorithms at all three time horizons (1, 5 and 10-step-ahead). The results of pairwise comparison tests ($p < 0.05$) among time series forecasting algorithms are reported in Table 1.

The mean number of features used by the forecasting algorithms is presented in Fig. 4.

The overall computational costs of the proposed and conventional forecasting techniques are summarized and compared in Fig. 5 and detailed in appendix table F6.

Key Findings:

- GEFTS significantly outperformed conventional algorithms both at short horizons (one-step ahead), and at longer horizons (5 and 10-step-ahead; Figs. 2–3, and Table 1). Jacob's Ensemble (JE), Homogeneous Ensemble of learners with static weighted voting approach (HOW), ARIMA–GARCH (A–G) and GRNN are the other top rated forecasting algorithms.
- It appears that GEFTS performs better in situations where the time series are not deseasonalized (Table 1). Not just GEFTS, but we also observe the similar trend among most of the other homogeneous ensemble approaches—HOW and HOS (Homogeneous Ensemble with unweighted voting). A plausible explanation for this puzzle would be we likely introduced errors while deseasonalizing the data. On the other hand, for Jacob's homogeneous ensemble, apparently the

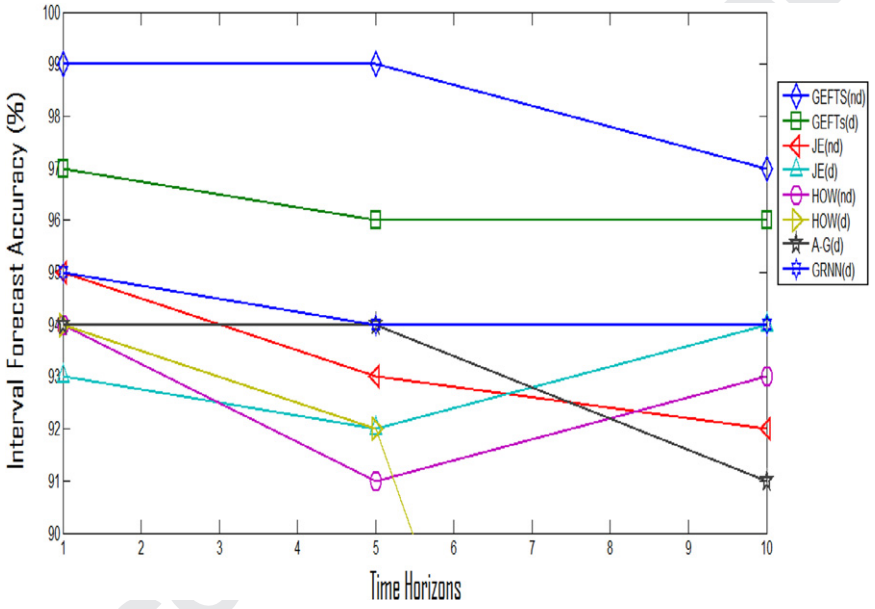

Fig. 2. A comparative study of interval forecasting accuracy.

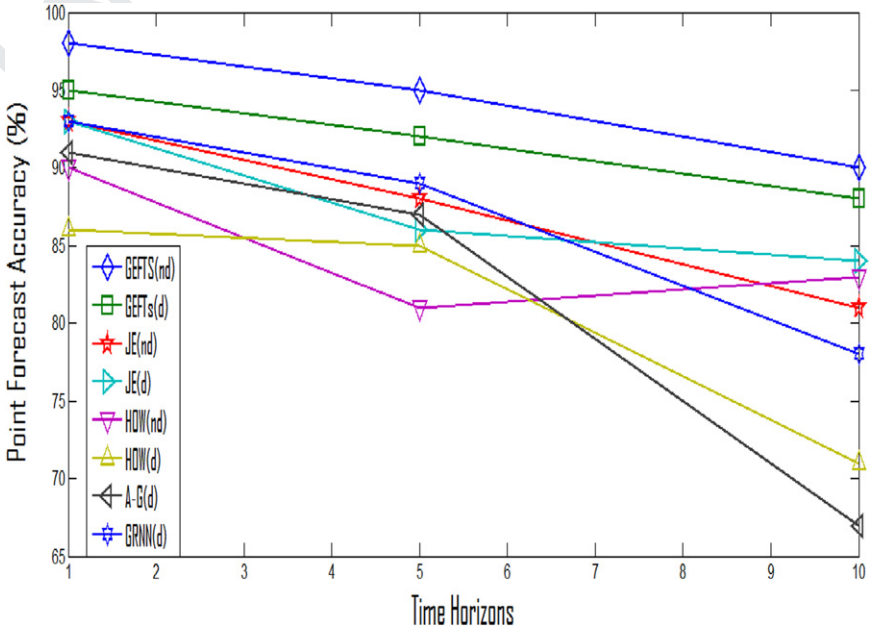

Fig. 3. A comparative study of point forecasting accuracy.

Please cite this article as: I.A. Gheyas, L.S. Smith, A novel neural network ensemble architecture for time series forecasting, Neurocomputing (2011), doi:10.1016/j.neucom.2011.08.005

Table 1
Pairwise comparisons between time series forecasting algorithms.

Rank	Algorithms	Significantly outperformed algorithms
1-step-ahead forecast		
1	GEFTS(nd)	(1)GEFTS (d), (2)A–G(d), (3)JE (nd), (4)JE(d), (4)GRNN(d), (5)HOW(nd), (6)SVM (d), (7)HOW (d), (8)HOS (nd), (9)HEW(d), (10)HOS (d), (11)HES (d), (12)MLP (d), (13) HES (nd), (14)ERNN(d), (15)RBFN (d), (16) HA(d), (17) HEW(nd), (18)SVM (nd), (19)GRNN(nd), (20)RBFN(nd), (21)MLP (nd), (22)ERNN(nd), (23) HA (nd), (24) A–G (nd)
2	GEFTS(d)	(1)A–G(d), (2)JE (nd), (3)JE(d), (4)GRNN(d), (5)HOW(nd), (6)SVM (d), (7)HOW (d), (8)HOS (nd), (9)HEW(d), (10)HOS (d), (11)HES (d), (12)MLP (d), (13) HES (nd), (14)ERNN(d), (15)RBFN (d), (16) HA(d), (17) HEW(nd), (18)SVM (nd), (19)GRNN(nd), (20)RBFN(nd), (21)MLP (nd), (22)ERNN(nd), (23) HA (nd), (24) A–G (nd)
3	A–G (d), JE (d), JE (nd), GRNN(d)	(1)HOW(nd), (2)SVM (d), (3)HOW (d), (4)HOS (nd), (5)HEW(d), (6)HOS (d), (7)HES (d), (8)MLP (d), (9) HES (nd), (10)ERNN(d), (15)RBFN (d), (11) HA(d), (12) HEW(nd), (13)SVM (nd), (14)GRNN(nd), (15)RBFN(nd), (16)MLP (nd), (17)ERNN(nd), (18) HA (nd), (19) A–G (nd)
4	HOW (nd), SVM (d)	(1)HOW (d), (2)HOS (nd), (3)HEW(d), (4)HOS (d), (5)HES (d), (8)MLP (d), (6) HES (nd), (7)ERNN(d), (8)RBFN (d), (9) HA(d), (10) HEW(nd), (11)SVM (nd), (12)GRNN(nd), (13)RBFN(nd), (14)MLP (nd), (15)ERNN(nd), (16) HA (nd), (17) A–G (nd)
5	HOW (d), HOS (nd)	(1)HEW(d), (2)HOS (d), (3)HES (d), (4)MLP (d), (5) HES (nd), (6)ERNN(d), (7)RBFN (d), (8) HA(d), (9) HEW(nd), (10)SVM (nd), (11)GRNN(nd), (12)RBFN(nd), (13)MLP (nd), (14)ERNN(nd), (15) HA (nd), (16) A–G (nd)
6	HEW(d), HOS(d)	(1)HES (d), (2)MLP (d), (3) HES (nd), (4)ERNN(d), (5)RBFN (d), (6) HA(d), (7) HEW(nd), (8)SVM (nd), (9)GRNN(nd), (10)RBFN(nd), (11)MLP (nd), (12)ERNN(nd), (13) HA (nd), (14) A–G (nd)
7	HES(d), MLP(d), HES(nd), ERNN(d), RBFN(d), HA(d)	(1) HEW(nd), (2)SVM (nd), (3)GRNN(nd), (4)RBFN(nd), (5)MLP (nd), (6)ERNN(nd), (7) HA (nd), (8) A–G (nd)
8	HEW(nd), SVM(nd), GRNN(nd), RBFN(nd)	(1)MLP (nd), (2)ERNN(nd), (3) HA (nd), (4) A–G (nd)
9	MLP(nd), ERNN (nd), HA(nd), A–G(nd)	
5-step-ahead forecast		
1	GEFTS(nd)	(1)GEFTS(d), (2)A–G(d), (3)GRNN(d), (4)HOW(d), (5)ERNN (d), (6)JE(d), (7)JE(nd), (8)HOW(nd), (9)HOS(nd), (10)SVM(d), (11)HEW(d), (12)HOS(d), (13) MLP(d), (14) RBFN(d), (15) HA (d), (16)HEW (nd), (17)HES(nd), (18)HES(d), (19)GRNN (nd), (20)RBFN(nd), (21) MLP(nd), (22)ERNN (nd), (23)HA(nd), (24)SVM(nd), (25)A–G(nd)
2	GEFTS(d)	(1)A–G(d), (2)GRNN(d), (3)HOW(d), (4)ERNN (d), (5)JE(d), (6)JE(nd), (7)HOW(nd), (8)HOS(nd), (9)SVM(d), (10)HEW(d), (11)HOS(d), (12) MLP(d), (13) RBFN(d), (14) HA (d), (15)HEW (nd), (16)HES(nd), (17)HES(d), (18)GRNN (nd), (19)RBFN(nd), (20) MLP(nd), (21)ERNN (nd), (22)HA(nd), (23)SVM(nd), (24)A–G(nd)
3	A–G(d), GRNN(d)	(1)HOW(d), (2)ERNN (d), (3)JE(d), (4)JE(nd), (5)HOW(nd), (6)HOS(nd), (7)SVM (d), (8)HEW(d), (9)HOS(d), (10) MLP(d), (11) RBFN(d), (12) HA (d), (13)HEW (nd), (14)HES(nd), (15)HES(d), (16)GRNN (nd), (17)RBFN(nd), (18) MLP(nd), (19)ERNN (nd), (20)HA(nd), (21)SVM(nd), (22)A–G(nd)
4	HOW(d), ERNN (d), JE (d), JE(nd)	(1)HOW(nd), (2)HOS(nd), (3)SVM (d), (4)HEW(d), (5)HOS(d), (6) MLP(d), (7) RBFN(d), (8) HA (d), (9)HEW (nd), (10)HES(nd), (11)HES(d), (12)GRNN (nd), (13)RBFN(nd), (14) MLP(nd), (15)ERNN (nd), (16)HA(nd), (17)SVM(nd), (18)A–G(nd)
5	HOW(nd), HOS(nd), SVM(d)	1)HEW(d), (2)HOS(d), (3) MLP(d), (4) RBFN(d), (5) HA (d), (6)HEW (nd), (7)HES(nd), (8)HES(d), (9)GRNN (nd), (10)RBFN(nd), (11) MLP(nd), (12)ERNN (nd), (13)HA(nd), (14)SVM(nd), (15)A–G(nd)
6	HEW(d), HOS(d), MLP(d), RBFN(d), HA(d)	1)HEW (nd), (2)HES(nd), (3)HES(d), (4)GRNN (nd), (5)RBFN(nd), (6) MLP(nd), (7)ERNN (nd), (8)HA(nd), (9)SVM(nd), (10)A–G(nd)
7	HEW(nd), HES(nd), HES(d), GRNN(nd), RBFN(nd), MLP(nd), ERNN(nd), HA(nd), SVM(nd), A–G(nd)	
10-step-ahead forecast		
1	GEFTS (nd)	(1)GEFTS(d), (2)HOW(nd), (3)JE(nd), (4)JE(d), (5)HOS(nd), (6) HOS(d), (7)GRNN(d) (8) HEW(d), (9)HES(d), (10)HOW(d), (11)HEW(nd), (12)HES(nd), (13)MLP(d), (14) A–G(d), (15)ERNN(d), (16)RBFN(d), (17) SVM(d), (18)HA(d), (19)GRNN(nd), (20)MLP(nd), (21)ERNN(nd), (22)RBFN(nd), (23)SVM(nd), (24)A–G(nd), (25)HA(nd)
2	GEFTS (d)	1)HOW(nd), (2)JE(nd), (3)JE(d), (4)HOS(nd), (5) HOS(d), (6)GRNN (d), (7) HEW(d), (8)HES(d), (9)HOW(d), (10)HEW(nd), (11)HES(nd), (12)MLP(d), (13) A–G(d), (14)ERNN(d), (15)RBFN(d), (16) SVM(d), (17)HA(d), (18)GRNN(nd), (19)MLP(nd), (20)ERNN(nd), (21)RBFN(nd), (22)SVM(nd), (23)A–G(nd), (24)HA(nd)
3	HOW(nd), JE(d), JE(nd)	(1)HOS(nd), (2) HOS(d), (3)GRNN (d), (4) HEW(d), (5)HES(d), (6)HOW(d), (7)HEW(nd), (8)HES(nd), (9)MLP(d), (10) A–G(d), (11)ERNN(d), (12)RBFN(d), (13) SVM(d), (14)HA(d), (15)GRNN(nd), (16)MLP(nd), (17)ERNN(nd), (18)RBFN(nd), (19)SVM(nd), (20)A–G(nd), (21)HA(nd)
4	HOS(nd), HOS(d), GRNN(d), HEW(d), HES(d)	(1)HOW(d), (2)HEW(nd), (3)HES(nd), (4)MLP(d), (5) A–G(d), (6)ERNN(d), (7)RBFN(d), (8) SVM(d), (9)HA(d), (10)GRNN(nd), (11)MLP(nd), (12)ERNN(nd), (13)RBFN(nd), (14)SVM(nd), (15)A–G(nd), (16)HA(nd)
5	HOW(d), HEW(nd), HES(nd)	(1)MLP(d), (2) A–G(d), (3)ERNN(d), (4)RBFN(d), (5) SVM(d), (6)HA(d), (7)GRNN(nd), (8)MLP(nd), (11)ERNN(nd), (12)RBFN(nd), (13)SVM(nd), (14)A–G(nd), (15)HA(nd)
6	MLP(d), A–G(d), ERNN(d), RBFN(d), SVM(d)	(1)HA(d), (2)GRNN(nd), (3)MLP(nd), (4)ERNN(nd), (5)RBFN(nd), (6)SVM(nd), (7)A–G(nd), (8)HA(nd)
7	HA(d), GRNN(nd), MLP(nd), ERNN(nd), RBFN(nd), SVM(nd)	(1) A–G(nd), (2) HA(nd)
8	A–G(nd), HA(nd)	

deseasonalization offers no significant performance improvement (no statistically significant differences were found before and after the deseasonalization). All the single neural networks (GRNN, ERNN, MLP and RBFN), ARIMA–GARCH, SVM and HA achieve improved performance when applied to deasonalized data.

- It was notable that for non-deseasonalized time series the overall best forecasting performances came from those algorithms that usually use a fairly large number of features (lagged variables) and these algorithms all are homogeneous ensembles (Fig. 4). In this context, it is important to note that the performance of each forecasting algorithms was assessed on unseen data.

Please cite this article as: I.A. Gheyas, L.S. Smith, A novel neural network ensemble architecture for time series forecasting, Neurocomputing (2011), doi:10.1016/j.neucom.2011.08.005

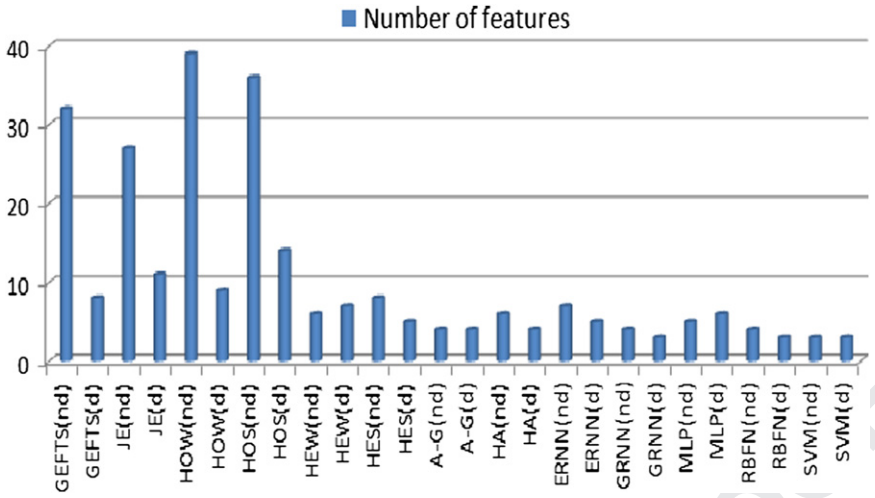

Fig. 4. A comparative study of the average number of features included in the analysis.

- All forecasting algorithms are computationally expensive (Fig. 5). This is because in all forecasting algorithms in our comparison, we used SAGA and PSO to optimize parameters. In model building, there is always a trade off between the accuracy and computational cost.

5. Summary and conclusions

We present an effective algorithm (GEFTS) for univariate time series forecasting. Our proposed algorithm is a hybrid of well-established algorithms. Our study suggests that sophisticated algorithms together can provide additional competitive advantage, which cannot be achieved by applying single time series forecasting algorithm only. GEFTS is an ensemble algorithm wherein both the base learners and the combiner are GRNN learners. We compare our proposed algorithm with the best known algorithms on 30 real datasets. The one-step process is iterated to obtain multi-step-ahead forecasts. The results obtained from experiments show that although GEFTS is an amalgamation of existing algorithms, it is superior to conventional algorithms, both for short-term and long-term out-of-sample forecasting. The only downside we can see about GEFTS is its rather high computational cost. However, accuracy is more important; speed without accuracy is no good.

Acknowledgements

We thank the reviewers and editorial board members of the Journal of Neurocomputing for strengthening our paper.

Appendix A: Generalized regression neural Networks (GRNN)

GRNN is a simple, yet very powerful learning algorithm. GRNN is an instance-based algorithm. In GRNN each observation in the training set forms its own cluster. When a new input pattern $x=(x_1,\ldots,x_n)$ is presented to the GRNN for the prediction of the output value, each training pattern (prototype pattern) $y_i=(y_{i1},\ldots,y_{in})$ assigns a membership value h_i to x based on the Euclidean distance d, where

$$d=d(x,y_i)=\sqrt{\sum_{j=1}^{m}(x_j-y_{ij})^2} \tag{A1}$$

and

$$h_i=\exp\left(-\frac{d^2}{2\sigma^2}\right) \tag{A2}$$

n is the total number of features in the study. x_j is the value of the jth feature of the presented pattern (features can be multivalued or not). y_{ij} is the value of the jth feature of the ith prototype pattern and σ is the smoothing function parameter. We found that the performance of GRNN is not very sensitive to the exact setting of the parameter (σ). We arbitrarily set each centre's width to 0.5 times of the average distance to 20 nearest neighbours.

Finally, GRNN calculates the output value z of the pattern x as in Eq. (B2). The predicted output of the GRNN for the pattern x is the weighted average of the outputs of all prototype patterns. GRNN can handle continuous output variables and categorical output variables with two categories: event of interest (coded as '1') or not (coded as '0'):

$$z=\frac{\sum_i(h_i\times \text{output of } y_i)}{\sum_i h_i} \tag{A3}$$

If the output variable is binary, then GRNN calculates the probability of event of interest. If the output variable is continuous, then it estimates the value of the variable.

Appendix B: Overview of SAGA

The proposed algorithm GEFTS uses an improved algorithm SAGA [6] for selecting an optimal subset of features, both for base GRNNs and the combiner GRNN. SAGA uses GRNN for assessing the fitness of feature subsets. SAGA works in three stages. During stage 1, SAGA applies the Simulated Annealing algorithm (SA) on 100 randomly selected possible solutions. SA leads to global exploration of search space without getting trapped into a local minimum. If the best solution does not improve 100 consecutive generations, the first stage is terminated. During stage 2, SAGA applies the Genetic Algorithm (GA) on the 100 best-to-date solutions found by the SA. A total of 50 pairs are picked from the chromosome pool using linear ranking selection. Selection is done "with replacement" meaning that the same chromosome can be selected more than once to become a parent. Each pair creates two offspring using the half uniform crossover scheme (HUX) and then the parents die. In HUX, exactly half of the non-matching parent's genes are swapped. Due to selection of the fittest chromosomes, the crossover and a very low mutation

Please cite this article as: I.A. Gheyas, L.S. Smith, A novel neural network ensemble architecture for time series forecasting, Neurocomputing (2011), doi:10.1016/j.neucom.2011.08.005

rate (0.0001), GA converges quickly to a near optimal solution. The second stage ends if the best solution does not improve in 100 consecutive generations. In the final stage, SAGA refines the search by hill-climbing on the best-to-date solution. The pseudo codes of SA, GA and HC are given below.

We encode possible feature subset solutions in ordered, fixed-length binary strings where '1' indicates the presence of the feature and '0' its absence.

//Pseudo code of SA

Step 1 Initialize parameters:

- Set the initial temperature (T_i): T_i=Maximum number of iterations for SA.
- Set the current temperature (T_c): $T_c=T_i$
- Randomly select 100 feature subset solution $I(=I(1{:}100))$ from the pool of possible solutions for initial population.

Step 2 Evaluate the fitness of each solution: Measure the fitness $E_o(=E_o(1{:}100))$ of solutions in the population using GRNN and store the information (feature subset solutions with fitness scores) where, $0 \le E_o \le 1$. For the method of fitness (E_o) assessment, see Section 2.2.2.

Step 3 Update the effective temperature (T_c):

$$T_c = T_c - T_{spent} \tag{B1}$$

where T_{spent}=total time spent so far.

Step 4: For all current feature subset vectors $I(=I(1{:}100))$ change the bits of vectors with probability $p_{mu}(=p_{mu}(1{:}100))$.

Step 5: Evaluate the fitness $E_n(=E_n(1{:}100))$ of the new candidate solutions if not already evaluated.

Step 6 Determine if this new solution is kept or rejected and update the database:

If $E_n \ge E_o$, the new solution is accepted. The new solution replaces the old solution and E_o is set to E_n:$E_o=E_n$. Else we accept the new solution with a certain probability

$$p_{ac}(=p_{ac}(1:100)) : p_{ac} = \exp\left(-\frac{(E_o - E_n)}{T_c}\right). \tag{B2}$$

Step 7: Update the effective temperature T_c. If the effective temperature is greater than zero, return to step 4. Otherwise, the run is finished.

//The pseudo code of GA

Step 1: Construct a chromosome pool of size 100 with the 100 fittest chromosomes from the list of feature subset solutions evaluated so far by the SA.

Step 2: Select 50 pairs of chromosomes with replacement using rank-based selection strategy.

Step 3: Perform crossover between the chromosomes using the half uniform crossover scheme (HUX). In HUX, half of the non-matching parents' genes are swapped.

Step 4: Kill the parent solutions.

Step 5: Mutate offspring with probability 0.0001.

Step 6: Evaluate the fitness of the offspring provided if it has not already been evaluated and if sufficient time is available. Update the database and estimate the time left.

Step 7: Go back to step 2 if the time is not up.

//Pseudo code of Hill-climbing algorithm

Step 1: Select the best-to-date solution.

Step 2: Create 10,000 new candidate solutions from the selected solution by changing only one bit (feature) at a time.

Step 3: Evaluate the new solutions if they are not evaluated before and update the database. Replace the previous solution by the new solution(s) if they are better than the previous solution.

Step 4: Go back to step 2 and perform the hill climbing on each of the accepted new solutions. Repeatedly apply the process from steps 2 to 3 on selected solutions as long as the process is successful in finding improved solutions in every repetition and as long as the time is available.

Table C1
The performance of different smoothing factor parameter values.

Rank (based on statistical significance)	Each centre's width	Classification accuracy (%)
4	Euclidean distance to the nearest member	79 (12)
3	(Euclidean distance to the nearest member)∗0.1	85 (8)
3	(Euclidean distance to the nearest member)∗0.2	87 (7)
3	(Euclidean distance to the nearest member)∗0.3	88 (7)
3	(Euclidean distance to the nearest member)∗0.4	84 (8)
4	(Euclidean distance to the nearest member)∗0.5	81 (12)
1	(Average Euclidean distance to 5 nearest members)∗0.5	95 (4)
1	(Average Euclidean distance to 10 nearest members)∗0.5	96 (4)
1	(Average Euclidean distance to 20 nearest members)∗0.5	97 (3)
1	(Average Euclidean distance to 30 nearest members)∗0.5	96 (5)
1	(Average Euclidean distance to 40 nearest members)∗0.5	95 (5)
2	(Average Euclidean distance to 50 nearest members)∗0.5	92 (7)
3	(Average Euclidean distance to 60 nearest members)∗0.5	88 (10)
4	(Average Euclidean distance to 70 nearest members)∗0.5	84 (11)
5	(Average Euclidean distance to 20 nearest members)∗1.5	75 (12)
5	(Average Euclidean distance to 90 nearest members)∗2	72 (14)
5	(Average Euclidean distance to 100 nearest members)∗2.5	72 (17)

Appendix C

See Table C1

References

[1] S. Pellegrini, E. Ruiz, A. Espasa, Prediction intervals in conditionally heteroscedastic time series with stochastic components, International Institute of Forecasters 27 (2011) 308–319.
[2] A.K. Dhamija, V.K. Bhalla, Financial time series forecasting: comparison of neural networks and ARCH models, International Research Journal of Finance and Economics 49 (2010) 185–202.
[3] Z. Zhang, C. Shi, S. Zhang, Z. Shi, Stock time series forecasting using support vector machines employing analyst recommendations, Lecture Notes in Computer Science 3973 (2006) 452–457.
[4] D.F. Specht, A general regression neural network, IEEE Transactions on Neural Networks 20 (1991) 568–576.
[5] I. Eggleton, Intuitive time series extrapolation, Journal of Accounting Research (Supplement) (1976) 68–131.
[6] I.A. Gheyas, L.S. Smith, Feature subset selection in large dimentionality domains, Pattern Recognition 43 (2010) 5–13.
[7] H.K. Cigizoglu, M. Alp, Generalized regression neural network in modelling river sediment yield, Advances in Engineering Software 37 (2006) 63–68.
[8] M. Shiblee, P.K. Kalra, B. Chandra, Time series prediction with multilayer perceptron (MLP): a new generalized error based approach, Lecture Notes in Computer Science 5507 (2009) 37–44.
[9] D.N. Kumar, K.S. Raju, T. Sathish, River flow forecasting using recurrent neural networks, Water Resources Management 8 (2004) 143–161.
[10] X. Yu, J. Zhang, A comparison of Hybrid ARMA-Elman models with single models for forecasting interest rates, in: Proceedings of the Second International Symposium on Intelligent Information Technology Application 2(2008) 985–989.
[11] V. Barrile, M. Cacciola, S. D'Amico, A. Greco, F.C. Morabito, F. Parrillo, Radial basis function neural networks to foresee aftershocks in seismic sequences related to large earthquakes, Lecture Notes in Computer Science 4233 (2006) 909–916.

Please cite this article as: I.A. Gheyas, L.S. Smith, A novel neural network ensemble architecture for time series forecasting, Neurocomputing (2011), doi:10.1016/j.neucom.2011.08.005

[12] J.P. Pabico, E.R.E. Mojica, J.R.L. Micor, Design of homogenous ensemble for splice-site indentification in human sequences, in: Proceedings of the Tenth International Conference on Molecular System Biology, University of the Philippines Diliman, 25–28 February 2008, pp. 60–62.
[13] Z. Shen, F. Kong, Optimizing weights by genetic algorithm for neural network ensemble, Lecture Notes in Computer Science 3173 (2004) 323–331.
[14] C. Tsai, Y. Lin, D.C. Yen, Y. Chen, Predicting stock returns by classifier ensembles, Applied Soft Computing 11 (2011) 2452–2459.
[15] G. Tsomakas, L. Angelis, I. Vlahavas, Selective fusion of heterogeneous classifiers, Intelligent Data Analysis 9 (2005) 1571–4148.
[16] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of experts, Neural Computation 3 (1991) 79–87.
[17] S. Siegel, N.J. Castellan Jr, Nonparametric Statistics: For the Behavioural Sciences, Second ed., McGraw-Hill, New York, 1988.
[18] Economagic.com: Economic Time Series Page. [Online]. Available at: 〈http://www.economagic.com/fedbog.htm〉.

Iffat A. Gheyas received a Ph.D. from the University of Stirling in the area of Machine Learning in 2010. Between 2010 and 2011, she worked as a KTP Associate on a joint research project between Time for Medicine Limited and the University of Glamorgan to develop optimum neural network models for medical applications. She is currently appointed as a research fellow in the University of Aberdeen Business School.

Leslie S. Smith received the B.Sc degree in 1973, and the Ph.D. in 1981, both from Glasgow University. From 1980 to 1983, he was a lecturer at Glasgow University. Since 1984 he has worked at the Stirling University, where he is now a Professor of Computing Science and Head of the Department of Computing Science and Mathematics. His research interests are in signal processing for neural systems, engineering approximations to early auditory processing, neural/electronic interfacing and neuromorphic systems. He is a Senior Member of the IEEE and member of the Acoustical Society of America and the Society for Neuroscience.

Please cite this article as: I.A. Gheyas, L.S. Smith, A novel neural network ensemble architecture for time series forecasting, Neurocomputing (2011), doi:10.1016/j.neucom.2011.08.005