NOTICE: this is the author's version of a work that was accepted for publication in Neurocomputing. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Neurocomputing, [VOL 135, July 2014] DOI# 10.1016/j.neucom.2013.12.037

The final paper can be found here
(<http://www.sciencedirect.com/science/article/pii/S0925231214000551>)

A Neural Gas Mixture Autoregressive Network for Modelling and Forecasting FX Time Series

Yicun Ouyang, Hujun Yin*

School of Electrical and Electronic Engineering, University of Manchester, Manchester, M60 1QD, UK

Abstract

Nowadays, there exist various methods for modelling and forecasting foreign exchange (FX) rates including economical models, statistical methods and learning neural networks. Dealing with the problems of nonstationarity and nonlinearity has been a challenge. In this paper, we propose a combined neural model for effectively tackling the problems. The model is termed as neural gas mixture of autoregressive models (NGMAR) and it organises the mixture of autoregressive models in the way of the neural gas. By taking the advantages of dynamic neighbourhood rankings of neural gas and the more appropriate similarity measure of the sum of autocorrelation coefficients, the model is able to effectively model and forecast nonstationary and nonlinear time series. The NGMAR has been tested on several benchmark data sets as well as a variety of FX rates. The experimental results show that the proposed method outperforms significantly other methods, in terms of normalised root mean squared error and correct trend prediction percentage.

Keywords:

Modelling and forecasting FX rates, neural gas, mixture autoregressive model.

1. Introduction

The foreign exchange (FX) market is the largest and most lucrative financial market with an estimated \$1 trillion traded a day [16]. Exchange rates are among the most important economic indices [39]. The decision makers of large companies with international business have the very necessity for accurately forecasting exchange rates while transferring substantial amount of currencies for their business to make as much profit as possible. However, the exchange rates are highly volatile and affected by many economic, political and even psychological factors [16]. These factors interact in a complex fashion, thus predicting FX rates is deemed as a difficult and challenging task. Many researchers have been engaging in developing methods and analysis tools.

Financial time series analysis has relied on time series models for a long time. Simple econometric models, which provide a natural framework to study the structures of financial time series, have been exercised for many years. These models include autoregressive (AR), moving-average (MA), autoregressive moving-average (ARMA) models and so on. They all try to build a linear relationship among the past points. However, little evidence shows that linear dependence exists in exchange rate series [16] and non-linearity should be considered as the core of the analysis. Therefore, much more efforts have been devoted recently to exploring nonlinear models. Autoregressive conditional heteroscedastic (ARCH) [12] and generalized

autoregressive conditional heteroscedastic (GARCH) [2] models have been proposed for describing behaviours of volatility, but both violate a well-known fact that the price of a financial asset responds differently to positive and negative shocks [35]. Furthermore, a negative correlation between volatilities of current and future returns was found but not considered in these two models, though some attempts exist to address these drawbacks [25].

As flexible approximators and learning systems, neural networks have attracted growing interest in using them for modelling and forecasting time series. Since the first attempt [20], they have been demonstrated to be a competitive alternative to various traditional time series models [41]. The multilayer perceptrons (MLP) [11] [15] can be used for time series prediction due to its general universal approximation capabilities. Another type of feedforward neural network was proposed for modelling and forecasting univariate nonlinear and multivariate time series [4] [10]. A stochastic neural network based on the Kalman filter, featuring a multilayer feedforward architecture with random connections, was also introduced for nonlinear statistical prediction [28]. As it is often difficult to choose the architecture of the feedforward neural network, a systematic approach [7] was proposed to eliminate weights that are non-significant, thus simplifying the structure. A neural network support vector machine (SVM) was also implemented for time series prediction [32] [33] [36]. A Bayesian neural network, trained by a Monte Carlo algorithm, was applied to time series analysis and its generalization ability has been also explored and revealed to some extent [22]. In addition, the echo state network (ESN) [18], a recurrent neural network (RNN) with a sparsely connected hid-

*Corresponding author

Email addresses: yicun.ouyang@postgrad.manchester.ac.uk
(Yicun Ouyang), hujun.yin@manchester.ac.uk (Hujun Yin)

den layer, was reported working well in the NN3 forecasting competition [8].

The self-organising map (SOM) has been applied to modelling time series in different ways [31], of which one is dividing or clustering time series into segments. A SVM expert system [3] and the RNN [6] based on SOM-clustered local models were used for predicting time series. However, these models may not reach joint optimization due to their two-stages respective optimizing.

The SOM-based models are flexible in the sense that it can be successfully applied not only to the unsupervised learning tasks such as data clustering and visualization, but also to the supervised learning task function approximation. In the basic approach, consecutive time points are grouped into segments of fixed length, and then used as the input vectors for training the SOM [26]. This method is also termed as vector SOM (VSOM) [26]. To integrate temporal information into the map, the SOM was naturally extended by using recurrent connections and explicit representations [30]. The temporal Kohonen map (TKM) [5] was based on the standard SOM, further introducing previous activations in computing the current activation. The concept of integration of the activation was also applied to the recurrent SOM (RSOM) [37] [14] in the similar way as in the TKM, except for different expressions. Recursive SOM (RecSOM) [38], regarding the activation as another input homogeneous to the original input, was proposed to better capture the temporal information. It incorporated the context information into the weights by calculating the difference between previous activities and the weights of the corresponding map.

A mixture regressive model named self-organising mixture autoregressive (SOMAR) model [26] has been proposed recently. It has a different partition approach and similarity measure to reflect the characteristics of homogeneous time series. It was demonstrated that the SOMAR obtained the improved performance in modelling and predicting on a variety of time series than regressive and other SOM-based models.

The rest of this paper is organised as follows. In the next section, the regressive models and neural networks are briefly introduced. Section 3 describes the proposed model in details. Experiments and results are presented and discussed in section 4. The final section concludes the study.

2. Related Work

In the literature, there are a large number of publications relating to modelling and forecasting time series. They can be divided into two categories: regressive models and neural networks. Typical examples from both are briefly described in the following two subsections.

2.1. Regressive Models

Regressive models have been widely used for modelling time series. Assume $\{x_t\}_{t=1}^N$ is a univariate time series of length N , the ARMA model [34] can be written as

$$x_t = \phi_0 + \sum_{i=1}^p \phi_i x_{t-i} - \sum_{i=1}^q \theta_i \epsilon_{t-i} + \epsilon_t \quad (1)$$

where ϕ_0 is a constant, $\phi_1, \phi_2, \dots, \phi_p$ and $\theta_1, \theta_2, \dots, \theta_q$ are the parameters of the model, ϵ_t is a white noise with mean zero and variance σ_ϵ^2 , and p and q are non-negative integers.

Note that the ARMA (p, q) model can be also expressed as

$$(1 - \sum_{i=1}^p \phi_i B^i) x_t = \phi_0 + (1 - \sum_{i=1}^q \theta_i B^i) \epsilon_t \quad (2)$$

where B is the backshift operator.

With a differencing step applied to the time series before the ARMA, the ARIMA [34] model was derived and can be described by

$$(1 - \sum_{i=1}^p \phi_i B^i)(1 - B)^d x_t = \phi_0 + (1 - \sum_{i=1}^q \theta_i B^i) \epsilon_t \quad (3)$$

where d is a non-negative integer.

Let $\{a_t\}_{t=1}^N$ denote the error or residual series. To incorporate the volatility, the GARCH [2] model is defined as follows:

$$a_t = \sigma_t \epsilon_t \quad (4)$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^m \alpha_i a_{t-i}^2 + \sum_{j=1}^s \beta_j \sigma_{t-j}^2 \quad (5)$$

where m and s are non-negative integers, ϵ_t is a sequence of independent and identical distributed random variables with zero mean and unit variance, $\alpha_0 > 0$, $\alpha_i \geq 0$, $\beta_i \geq 0$, and $\sum_{i=1}^{\max(m,s)} (\alpha_i + \beta_i) < 1$. What should be noted here is that $\alpha_i = 0$ for $i > m$ and $\beta_j = 0$ for $j > s$.

2.2. Neural Networks

MLP is a feedforward network trained by a supervised learning technique, e.g. the back-propagation [29]. It consists of multiple layers of nodes with activation functions which are differentiable and nonlinear. The activation function usually has two forms: logistic and hyperbolic tangent.

The back-propagation algorithm, deemed as the generalization of ubiquitous least mean square (LMS) algorithm, consists of two passes: forward pass and backward pass. In the forward pass, the input signals are presented to the network layer by layer, and finally the output signals are responded. All the synaptic weights of the network are fixed during this process. In the backward pass, the weights are adjusted as the error signals back propagate through the network.

RNN is a type of neural networks in which the connections among units form a directed cycle [6]. It has internal states of the network which can be used for supervising and processing dynamic temporal behaviours of the network. Therefore, RNN has its own internal memory to process the input sequences. This feature makes the network competitive when it is applied to tasks related to temporal information such as handwriting recognition and time series prediction.

The ESN is an RNN with a sparsely connected hidden layer [18]. The dynamic and output of the ESN can be described by the following equations

$$\mathbf{u}(t+1) = \text{sig}(\mathbf{W}_u \cdot \mathbf{u}(t) + \mathbf{W}_{in} \cdot \mathbf{x}(t)) \quad (6)$$

$$\mathbf{y}(t) = \mathbf{W}_{out}^T \cdot \mathbf{u}(t) \quad (7)$$

where $\mathbf{u}(t)$ is the reservoir vector in the hidden layer, $\mathbf{x}(t)$ and $\mathbf{y}(t)$ are the input and output vectors respectively, sig is the sigmoid activation function, $\mathbf{W}_{in}$, $\mathbf{W}_{out}$ and $\mathbf{W}_x$ are the weight matrices of the input, output and the internal connection of the reservoir, respectively. Note that only the weight matrix of the output vectors is updated during training.

Another type of neural networks commonly used in time series is the unsupervised SOM-based methods. The conventional SOM or VSOM groups the consecutive time series into segments of fixed length for training the SOM. To integrate the temporal information into the map, the standard SOM has been extended in TKM by introducing the previous activations to compute the current activation.

A modification of the TKM, RSOM, was proposed by moving the leaky integrators from the units outputs towards their entry. The activation $\mathbf{y}_i(t)$ of unit i is defined as follows:

$$\mathbf{y}_i(t) = (1 - \alpha)\mathbf{y}_i(t - 1) + \alpha(\mathbf{x}(t) - \mathbf{w}_i(t)) \quad (8)$$

where $\alpha \in [0, 1]$ is the leaking coefficient, $\mathbf{x}(t)$ is the input vector and $\mathbf{w}_i(t)$ is the weight vector of unit i . Consequently, the best matching unit (BMU) can be calculated by

$$BMU(t) = \arg \min_i \|\mathbf{y}_i(t)\| \quad (9)$$

and the weights of the map are updated according to

$$\mathbf{w}_i(t + 1) = \mathbf{w}_i(t) + r(t)N_{ib}(t)\mathbf{y}_i(t) \quad (10)$$

where $r(t)$ is the learning rate, and $N_{ib}(t)$ is the neighbourhood function of the neuron i and BMU.

The main difference between RSOM and TKM is that TKM integrates the direction of activity change during distance calculation, but RSOM also integrates this change while modifying the weights of the prototypes. This results in RSOM being more explicit in capturing temporal patterns in the inputs than the TKM. The potentials of the RSOM for temporal sequence processing tasks have been demonstrated [14].

The RecSOM is another temporal extension of the SOM, whose training is self referent with the direction that the map learns how to classify its own last activities [38]. That means each neuron i in the map is associated with two weight vectors $\mathbf{w}_i^x$ and $\mathbf{w}_i^y$, and the activation $\mathbf{y}_i(t)$ is adapted by

$$\mathbf{y}_i(t) = \exp(-\alpha\|\mathbf{x}(t) - \mathbf{w}_i^x\|^2 - \beta\|\mathbf{y}(t - 1) - \mathbf{w}_i^y\|^2) \quad (11)$$

where $\alpha > 0$ and $\beta > 0$ are constant coefficients. The BMU is selected with the maximum value of $\mathbf{y}_i(t)$. The learning algorithm is similar to the original SOM algorithm, and weight vectors $\mathbf{w}_i^x$ and $\mathbf{w}_i^y$ are adapted by

$$\mathbf{w}_i^x(t) = \mathbf{w}_i^x(t - 1) + r^x(t)N_{ib}(t)(\mathbf{x}(t) - \mathbf{w}_i^x) \quad (12)$$

$$\mathbf{w}_i^y(t) = \mathbf{w}_i^y(t - 1) + r^y(t)N_{ib}(t)(\mathbf{y}(t - 1) - \mathbf{w}_i^y) \quad (13)$$

where $r^x(t)$ and $r^y(t)$ are the learning rates.

Compared to the RSOM, the RecSOM model has the similar aim but different form of activation by incorporating context information into weight vectors. Furthermore, the exponential function, which is continuous and takes on values between zero and one, is used as the transfer function in the RecSOM to make the learning process stable. However, its performance is sensitive to the coefficients α and β , on which the impacts of present and past inputs depend. Moreover, the computational cost of RecSOM is also greatly increased because of incorporating extra dimensions to the weight vectors.

The SOMAR was introduced [26] to learn a mixture of AR processes by a self-organising procedure. Each node learns an AR model. A nonlinear time series is considered as a mixture of linear AR processes. Based on the principle that a stochastic process is characterized by white noise residuals, the SOMAR model employs the autocorrelation of errors instead of errors themselves to evaluate the similarity between the input vectors and the prototypes of the map.

As for each small batch of the input, the estimate of the autocorrelation coefficient of errors $R_i(k)$ at lag k ($k < m$) is obtained by

$$R_i(k) = \frac{1}{m\sigma^2} \sum_{l=0}^{m-k-1} (e_i(t - l) - \mu)(e_i(t + k - l) - \mu) \quad (14)$$

where $e_i(t) = x(t) - \mathbf{w}_i^T \mathbf{x}_{t-1}^{(p)}$ is the modelling error at node i , which is of length m with mean μ and variance σ^2 , p is the order of local AR model i . The sum of autocorrelation of coefficients (SAC) is used as the criterion for selecting the BMU for the input batch,

$$BMU(t) = \arg \min_i \left(\sum_{j=-m}^m \|R_i(j)\| \right), i = 1, 2, \dots, N \quad (15)$$

where i is the index of the local model and N is the total number of the local models. That is, BMU is the unit that has the minimum SAC value. Then the weights of the winner and its neighbours are updated by

$$\mathbf{w}_i(t + 1) = \mathbf{w}_i(t) + r(t)N_{ib}(t)e_i(t)\mathbf{x}(t) \quad (16)$$

SOMAR can correctly detect and uncover underlying autoregressive models by employing SAC for evaluating the similarity [26].

3. Neural Gas Mixture Autoregressive

The proposed method consists of two essential components, namely, neural gas and local mixture autoregressive, which are described in detail in the following two subsections.

3.1. Neural Gas

The NG [24] is a variant of the SOM. By introducing the neighbourhood ranking, the structure of the NG model is varied dynamically instead of a predefined lattice as in the SOM. In doing so, as a vector quantization algorithm has many local minima, the NG method can reach better optimum. The

weights of reference vectors of the NG are updated according to the ranking orders of the units within the neighbourhood. This means that not only the closest reference vector to the input vector but also the second closest, third closest and so on, are adapted according to the distance to the input vector. Assume that there are in total N reference vectors and i_0 denotes the index of the closest one, i_1 the index of the second closest, etc. and i_{N-1} the index of the most distant one to the input vector $\mathbf{x}(t)$, then the weights are adapted by

$$\mathbf{w}_{i_k}(t+1) = \mathbf{w}_{i_k}(t) + \epsilon \exp(-k/\lambda)(\mathbf{x}(t) - \mathbf{w}_{i_k}(t)) \quad (17)$$

where both the learning rate or adaption step size ϵ and neighbourhood range λ decay with time.

Compared to other clustering methods such as k -means, maximum-entropy and the SOM, the NG achieves a more robust convergence with faster speed and lower distortion error [24]. However, the price for this advantage is increased computational effort.

3.2. Neural Gas Mixture Autoregressive

In this section, the proposed neural gas mixture autoregressive (NGMAR) model is described in detail for modelling and forecasting time series. The basic idea of NGMAR is to organise the units in the way of the NG and each unit corresponds to an autoregressive model in the mixture, which is used for describing the nonlinear relationship between data points.

Mixture autoregressive (MAR) has the ability to handle cycles and conditional heteroscedasticity in time series and has been recently used in modelling and predicting nonstationary and nonlinear time series. As nonstationary time series can be considered as a mixture or weighted average of several stationary AR processes, it can be expressed as [26]

$$F(x_t|F_{t-1}) = \sum_{i=1}^N \beta_{(i,x)} \Phi(x_t - \phi_{i0} - \phi_{i1}x_{t-1} - \dots - \phi_{ip_i}x_{t-p_i}) \quad (18)$$

where F_{t-1} represents the information at time $t-1$, ϕ is the parameter, N is the number of AR processes, p_i is the order of AR process i , Φ refers to some distribution functions and $\beta_{(i,x)}$ is the weighting of the i -th local regressive model.

Similar to the SOMAR, a number of nodes in the NGMAR are used to capture local regressions. But, instead of using lattice structure, the nodes are in the NG form for better approximation. That is, the NGMAR model, is considered as a mixture of several AR processes. The input vectors are used in small batches, the modelling error of the input vector and local models represented by weight vectors is a discrete time series. Assuming that the size of batch is m , the input batch $\mathbf{x}(t)$ is defined as

$$\mathbf{x}(t) = \{x_1(t), x_2(t), \dots, x_m(t)\} \quad (19)$$

then the modelling error $\mathbf{e}_i(t)$ at node i is a series of length m with mean μ and variance σ^2 ,

$$\mathbf{e}_i(t) = \{e_{i1}(t), e_{i2}(t), \dots, e_{im}(t)\} \quad (20)$$

where $e_{ij}(t) = x_j(t) - \mathbf{w}_i^T \mathbf{x}_j^{(p)}(t-1)$, p and $\mathbf{w}_i$ are the order and parameters of local AR model i respectively. Then the autocorrelation coefficient $R_i(k)$ at lag $k(k < m)$ is obtained by equation (14).

To satisfy the sufficient condition of the stochastic process, the modelling error should be close to white noise if the modelling procedure is correct. Therefore, SAC is used as the similarity measure of input and reference vectors, and the BMU (winner) is selected with the minimum SAC, i.e. equation (15).

In the NGMAR model, the reference vectors are adapted according to their distance from the input batches and their rankings of the SAC values. The weight updating rule of the NGMAR is similar to that of the NG, except that input batches are used instead. The weights are adapted by

$$\mathbf{w}_{i_k}(t+1) = \mathbf{w}_{i_k}(t) + \epsilon \exp(-k/\lambda) \mathbf{e}_{i_k}(t) \mathbf{x}(t) \quad (21)$$

where $\mathbf{x}(t) = \{x_1(t), x_2(t), \dots, x_m(t)\}$ is the input batch. The learning rate ϵ and neighbourhood range λ are defined as follows

$$\epsilon = \epsilon_0 \left(\frac{\epsilon_N}{\epsilon_0} \right)^{\frac{n}{N}} \quad (22)$$

$$\lambda = \lambda_0 \left(\frac{\lambda_N}{\lambda_0} \right)^{\frac{n}{N}} \quad (23)$$

where ϵ_0 and λ_N are initial values, ϵ_N and ϵ_0 are final values, n is the time index of iteration, and N is the total number of iteration.

Algorithm 1 The training procedure of NGMAR

Require: small batches of input vectors from sample time series;

Ensure: the reference vectors representing for weights of neurons;

- 1: Build the input vectors $\mathbf{x}(t)$ from the training time series;
 - 2: Initialize the weight matrix $\mathbf{w}_i$, and specify training parameters learning rates l and training iterations N ;
 - 3: **for** $t = 1$ to N **do**
 - 4: **for** each i of all neurons **do**
 - 5: Compute individual error $err(t, i)$ of neuron i and input $\mathbf{x}(t)$ on Eq. (20);
 - 6: Compute the SAC value $SAC(i)$ and select the BMU on Eq. (15);
 - 7: **end for**
 - 8: Compute rank of neurons and build RANK
 - 9: **for** each i of all neurons **do**
 - 10: Find the position $r(i)$ of neuron i in RANK
 - 11: Update the weights $\mathbf{w}_i$ of neuron i on Eq. (21);
 - 12: **end for**
 - 13: **end for**
 - 14: **return** the weight matrix $\mathbf{w}_i$;
-

The training procedure of NGMAR is summarized in Algorithm 1. In NGMAR, the neurons are organised in neural gas, which means the neighbourhood topology is based on their distances from input vector. Each neuron is associated with a reference vector, representing a local AR model. For any batch

Table 1: Minimum prediction errors (NRMSE) and significance test of various adaptive models on Mackey-Glass data

Model	AIC	Mean	Std(10^{-3})	p
ARMA	-1.06e+3	0.0517	N/A	2.2e-8
GARCH	-1.08e+3	0.0463	N/A	5.6e-7
VSOM	-1.01e+3	0.0484	1.5287	2.1e-7
VQTAM ¹	-0.98e+3	0.0520	0.9354	1.3e-8
RSOM	-1.01e+3	0.0480	2.0219	7.7e-7
RecSOM	-1.02e+3	0.0470	1.5166	4.0e-7
NG	-1.06e+3	0.0430	0.7314	8.6e-7
NGMAR	-1.18e+3	0.0316	1.7355	N/A
SOAR	-1.01e+3	0.0483	1.9728	5.9e-7
SOMAR	-1.10e+3	0.0385	0.3317	2.4e-5
MLP	-1.04e+3	0.0435	1.8207	3.2e-7
ESN ²	-1.13e+3	0.0355	2.5665	0.0227

of input, the SAC value between it and each neuron is calculated and used for selecting the BMU. The NGMAR modelling involves selection of the parameters such as the number of the neurons, the dimension of the input vectors, the size of the batch of input data and the training iterations. These parameters have certain effects on the performance of the NGMAR model for forecasting time series, so they should be optimized via a validation procedure along. The details are given in the next section along with the experiments on specific time series.

4. Experimental Results and Discussions

We used three examples to evaluate the performance of the NGMAR model. The first two are the Mackey-Glass and Lorenz time series, and the third one is daily foreign exchange rates. For each case, the mean value, standard variance and p value of the t-test over five trials were obtained for comparison. The prediction performance was measured on the test sequences.

As for each model used in the experiments, its predictive ability was described through an out-of-sample prediction scheme. As reviewed in [9] [17], there are a number of measures of prediction accuracy. To compare forecasting errors of various models over different datasets, we used the normalized root mean squared error (NRMSE) described in [1] [19] [21]

$$NRMSE = \sqrt{\frac{(1/M) \sum_{t=1}^M (x(t) - \hat{x}(t))^2}{(1/M) \sum_{t=1}^M (x(t) - \bar{x})^2}} \quad (24)$$

where $\hat{x}(t)$ and $\bar{x}$ are respectively the predicted value and sample mean of the test time series, M is the length of the test sequence.

4.1. Mackey-Glass Data

The Mackey-Glass series has been widely used as benchmark data for testing nonlinear models in a variety of literatures [1].

Figure 1: Mackey-Glass time series with parameter $\alpha = 0.2$, $\beta = -0.1$ and $\delta = 17$.

Figure 2: Actual and predicted test series for the Mackey-Glass time series

This data set is generated from a dynamic system defined by the following differential equation

$$\frac{dx(t)}{dt} = \beta x(t) + \frac{\alpha x(t - \delta)}{1 + x(t - \delta)^{10}} \quad (25)$$

where $x(t)$ is the sample time series at time step t . Adopting $\alpha = 0.2$, $\beta = -0.1$ and $\delta = 17$, we generated a time series of length 2000 with a step size of 0.1. In the experiment, 90% of the overall points were used as the training set, and the remaining points as the test set. The orders of local AR model used for describing the data were selected as 11 according to the BIC result, so the consecutive series were grouped into input vectors with each having 12 values. A sample Mackey-Glass time series is plotted in Fig. 1.

The actual and predicted test series by the NGMAR are shown in Fig. 2. The errors between the target time series and the corresponding predicted values by various regressive models and neural models are given in Table 1. Moreover, the significance test for the performance of various models was also carried out. The p value of t-test was used to test the significance and p values are shown in Table 1 along with the errors.

In this experiment, the training iteration was set to 100000, which is fifty times of the length of the input time series, to make the training procedure reliable. Although using more neurons likely leads to smaller errors, the number of neurons of the SOM-based models was set to 100 to strike a balance between efficiency and accuracy. The size of input batch was set to 20 in the SOMAR and NGMAR models.

¹Code obtained from <http://www.deti.ufc.br/~guilherme/Codes/ng.m>

²Code obtained from <http://reservoir-computing.org/node/129>

Figure 3: Lorenz time series with parameters $\sigma = 10$, $\rho = 28$ and $\beta = 8/3$.

Figure 4: Actual and predicted test series for the Lorenz time series

The orders p and q of ARMA are both chosen as 2, and m and s of GARCH as 3 and 2 respectively. In RSOM, α is set as 0.6, and α and β of RecSOM as 0.25 and 0.05. The neurons of hidden and output layer of MLP are 5 and 2, and the size of the reservoir of ESN is 10×10 .

As can be seen from the table, the NGMAR gave the lowest error, which is significantly lower (i.e. p value is smaller than 0.01) than all the other models except for the ESN. However, the ESN's error is still some distance from that of the NGMAR. Other SOM based models except for SOMAR yielded similar performances along with the MLP. It was evident that the proposed NGMAR outperforms other neural models.

4.2. Lorenz Data

The Lorenz data was generated by a simplified mathematical model used for atmospheric convection [23]. The model is known as Lorenz equations

$$\frac{dx(t)}{dt} = \sigma(y(t) - x(t)) \quad (26)$$

$$\frac{dy(t)}{dt} = x(t)(\rho - z(t)) - y(t) \quad (27)$$

$$\frac{dz(t)}{dt} = x(t)y(t) - \beta z(t) \quad (28)$$

where $x(t)$, $y(t)$ and $z(t)$ are the values of time series at time t . A typical choice for the parameters are as $\sigma = 10$, $\rho = 28$ and $\beta = 8/3$. For these values, the system exhibits chaotic behaviour [21]. The data set is constructed with initial value $x(0) = 12$, $y(0) = 2$ and $z(0) = 9$, and the step size is chosen as 0.02. 90% of the 1000 points were used to train and the remaining points to test the models. A sample Lorenz data, $x(t)$, is plotted in Fig. 3.

The actual and predicted test series by NGMAR are shown in Fig. 4. The proposed NGMAR model was compared with various models, and the results, including prediction errors and significance tests, are presented in Table 2.

The parameters of the methods were set in the similar way as the previous experiment to balance between efficiency and accuracy. In each iteration, the input vectors were randomly selected across the entire training dataset. In the SOMAR and

Table 2: Minimum prediction errors (NRMSE) and significance test of various adaptive models on Lorenz data

Model	AIC	Mean	Std	p
ARMA	-1.10e+3	0.0464	N/A	2.5e-8
GARCH	-1.12e+3	0.0450	N/A	1.8e-8
VSOM	-1.02e+3	0.0467	1.0e-3	2.1e-9
VQTAM	-0.98e+3	0.0526	1.5e-3	1.3e-9
RSOM	-1.03e+3	0.0467	0.7e-3	1.1e-9
RecSOM	-1.04e+3	0.0454	1.5e-3	1.4e-8
NG	-1.06e+3	0.0432	0.5e-3	2.8e-9
NGMAR	-1.31e+3	0.0232	1.5e-3	N/A
SOAR	-1.03e+3	0.0461	1.8e-3	2.2e-8
SOMAR	-1.15e+3	0.0346	1.7e-3	3.4e-6
MLP	-1.08e+3	0.0410	2.5e-3	6.1e-7
ESN	-1.21e+3	0.0295	5.1e-3	0.0297

NGMAR model, the input data were small batches of input vectors.

The results are similar to those of the Mackey-Glass data. That is the NGMAR was the best performing model, giving significantly lower errors than all other models. Although the SOMAR and ESN were significantly better than regressive models, other SOM-based methods and MLP, they were still inferior to the NGMAR.

4.3. Foreign Exchange Rates

The foreign exchange rates used in the experiments contain the daily closing prices of exchange rates (British Pound against US Dollar, Euro and JP Yen) from 2001 to 2012 (retrieved from the PACIFIC exchange rate service). In total there are 3000 points, and the first 2700 points were used as the training set and the remaining points as the test set. Figs. 5-7 plot the closing daily prices of three main foreign exchange rates. The size of the sliding window for the input vectors was selected as 10 according to the AIC and BIC. The task was to perform one-step-ahead prediction.

The orders p and q of ARMA are 2 and 1, and m and s of GARCH are 2 and 3 respectively. In RSOM, α is set as 0.7, and α and β of RecSOM as 0.15 and 0.05. The neurons of hidden and output layer of MLP are 3 and 2, and the size of the reservoir of ESN is 10×10 .

Figure 5: GBP/USD daily closing prices (total data set)

Figure 6: GBP/EUR daily closing prices (total data set)

Figure 7: GBP/JPY daily closing prices (total data set)

Figure 8: Actual and predicted test series for GBP/USD time series

Figure 9: Actual and predicted test series for GBP/EUR time series

Figure 10: Actual and predicted test series for GBP/JPY time series

To compare the performance of various models in these financial time series, an additional measure has been used:

Correct prediction percentage (%): The correct prediction percentage (CPP) of the foreign exchange rate return ($\ln \frac{x_{t+1}}{x_t}$) was used to describe how many percentage of the prediction is made in the right direction by checking whether the predicted returns have the same sign as the actual returns [26] [40]. It is defined as follows:

$$CPP = \frac{\text{Number of Correct Direction Predictions}}{\text{Total Number of Predictions}} \quad (29)$$

By introducing a new similarity measure between the input and reference vectors and the neighbourhood ranking rule of neural networks, the NGMAR model is shown to outperform existing methods in the experiments of predicting FX rates. Figs. 8-10 show the one-step-ahead prediction results of the test sets by the NGMAR model.

The results of various models for predicting FX rates are presented in Tables 3-5. Both prediction errors and correct prediction percentages were employed as criteria. It can be seen that the NGMAR performed significantly better than all other methods, with the lowest errors and highest CPP percentages.

4.4. Sensitivity analysis

As discussed in the previous section, the new model involves in adopting new training and predicting strategies, including incorporating a new similarity measure and neighbourhood ranking rule. These strategies are introduced for improving the prediction accuracy of neural networks. We have demonstrated

Table 3: Minimum prediction errors (NRMSE) and correct prediction percentage (%) on foreign exchange rates GBP/USD

Model	NRMSE	CPP	Std	p
ARMA	0.1036	51.61	N/A	3.7e-6
GARCH	0.1015	51.82	N/A	8.4e-5
VSOM	0.1187	50.45	0.78	5.2e-7
VQTAM	0.1175	50.45	1.26	5.0e-6
RSOM	0.1253	50.42	1.07	2.1e-6
RecSOM	0.1276	50.38	1.28	5.1e-6
NG	0.1097	51.02	1.63	3.9e-5
NGMAR	0.0751	57.58	0.78	N/A
SOAR	0.1053	51.98	0.78	3.4e-6
SOMAR	0.0846	55.95	0.56	0.0052
MLP	0.1105	50.65	2.73	6.0e-4
ESN	0.1041	51.91	1.32	3.5e-5

Table 4: Minimum prediction errors (NRMSE) and correct prediction percentage (%) on foreign exchange rates GBP/EUR

Model	NRMSE	CPP	Std	p
ARMA	0.0631	50.32	N/A	0.0054
GARCH	0.0640	50.58	N/A	0.0125
VSOM	0.0624	50.32	2.06	0.0016
VQTAM	0.0587	50.56	3.13	0.0161
RSOM	0.0637	50.70	3.10	0.0180
RecSOM	0.0601	50.65	1.60	6.7e-4
NG	0.0565	51.30	1.92	0.0043
NGMAR	0.0387	54.95	0.80	N/A
SOAR	0.0611	50.56	0.55	7.6e-6
SOMAR	0.0452	53.11	1.12	0.0173
MLP	0.0689	50.22	1.27	1.1e-4
ESN	0.0625	50.31	1.69	5.4e-4

Table 5: Minimum prediction errors (NRMSE) and correct prediction percentage (%) on foreign exchange rates GBP/JPY

Model	NRMSE	CPP	Std	p
ARMA	0.0662	51.29	N/A	0.0047
GARCH	0.0587	51.85	N/A	0.0015
VSOM	0.0671	50.15	1.86	9.1e-4
VQTAM	0.0658	50.23	2.09	0.0020
RSOM	0.0675	50.88	1.67	0.0015
RecSOM	0.0633	50.78	1.05	8.6e-5
NG	0.0621	50.60	2.01	0.0027
NGMAR	0.0375	54.55	0.49	N/A
SOAR	0.0517	51.36	1.33	9.9e-4
SOMAR	0.0442	53.10	0.96	0.0167
MLP	0.0664	50.18	1.75	6.6e-4
ESN	0.0525	51.16	2.17	0.0092

that over five different time series, including benchmark and FX rates data, these strategies are reasonably effective for predicting time series. In this section, we investigated how the choice of parameters of the model affects the predicting performance.

To find out how sensitive the forecasting accuracy is to the parameters of the proposed neural network, the experiments

Table 6: Effects of parameter N on prediction performance of NGMAR

N	25	100	225	400
Mackey-Glass	0.0476	0.0316	0.0311	0.0307
Lorenz	0.0357	0.0232	0.0225	0.0218
GBP/USD	0.0921	0.0751	0.0743	0.0734
GBP/EUR	0.0570	0.0387	0.0379	0.0376
GBP/JPY	0.0551	0.0375	0.0372	0.0368

Table 7: Effects of parameter p on prediction performance of NGMAR

p	7	9	11	13
Mackey-Glass	0.0338	0.0324	0.0316	0.0341
Lorenz	0.0253	0.0248	0.0232	0.0239
GBP/USD	0.0773	0.0751	0.0757	0.0771
GBP/EUR	0.0404	0.0387	0.0391	0.0410
GBP/JPY	0.0382	0.0375	0.0378	0.0391

were performed by varying the number of neurons N and the order of local AR model p .

This sensitivity analysis resulted in a total of 10 experiments, where analysis of errors was performed on the NRMSE. Tables 6, 7 show the results of the experiments, and the conclusions of results can be summarised as follows

1) The prediction errors of test set, which are measured by NRMSE, are changing with parameters N and p .

2) Although some errors decrease as N increases, the differences are not significant when N is larger than 100. For example, the error difference between 100 and 400 on Mackey-Glass data is not more than 3%. Since the larger number of neurons consumes much more time, the balance is needed between accuracy and efficiency.

3) In Table 7, the optimal parameter p chosen according to AIC or BIC yields the best performance on the test set. This means that the parameter chosen based on AIC or BIC is consistent with the experiment results.

In terms of determining the number of neurons, there is another option called growing neural gas (GNG) [13] to construct neural gas. For GNG, it is not necessary to decide on the number of neurons because these neurons are added incrementally during execution. Insertion of new neurons stops only when the performance criteria is met or a maximum network size has been reached. A desirable insertion policy can be that a new node is inserted if the mean squared error is greater than a threshold.

5. Conclusions

In this paper, a new approach NGMAR has been proposed to model and forecast time series. It organises the local mixture autoregressive models as neural gas and uses them to describe the behaviour of time series. The experimental results show that this new combined neural model is more capable and accurate than regressive models, SOM-based models and other neural networks for predicting some benchmark data and FX rates series. They also show that the improvement by the NGMAR model is significant.

References

- [1] Barreto, G. A. and Araujo, A. F. R. (2004). Identification and control of dynamical systems using the self-organising map. *IEEE Transactions on Neural Networks* 15(5): 1244-1259.
- [2] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics* 31(3): 307-327.
- [3] Cao, L. (2003). Support vector machines experts for time series forecasting. *Neurocomputing* 51: 321-339.
- [4] Chakraborty, K., Mehrotra, K., Mohan, C. K. and Ranka, S. (1992). Forecasting the behavior of multivariate time series using neural networks. *Neural Networks* 5(6): 961-970.
- [5] Chappell, G. J. and Taylor, J. G. (1993). The temporal Kohonen map. *Neural Networks* 6(3): 441-445.
- [6] Cherif, A., Cardot, H. and Bon, R. (2011). SOM time series clustering and prediction with recurrent neural networks. *Neurocomputing* 74(11): 1936-1944.
- [7] Cottrell, M., Girard, B., Girard, Y., Mangeas, M. and Muller, C. (1995). Neural modeling for time series: A statistical stepwise method for weight elimination. *IEEE Transactions on Neural Networks* 6(6): 1355-1364.
- [8] Crone, S. F., Hibon, M. and Nikolopoulos, K. (2011). Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of Forecasting* 27(3): 635-660.
- [9] De Gooijer, J. G. and Hyndman, R. J. (2006). 25 years of time series forecasting. *International Journal of Forecasting* 22(3): 443-473.
- [10] De Groot, C. and Wrtz, D. (1991). Analysis of univariate time series with connectionist nets: A case study of two classical examples. *Neurocomputing* 3(4): 177-192.
- [11] Ebrahimpour, R., Nikoo, H., Masoudnia, S., Yousefi, M. R. and Ghaemi, M. S. (2011). Mixture of MLP-experts for trend forecasting of time series: A case study of the Tehran stock exchange. *International Journal of Forecasting* 27(3): 804-816.
- [12] Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica* 50(4): 987-1007.
- [13] Fritzke, B. (1995). A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems*, vol. 7, MIT Press: 625-632.
- [14] Giles, C. L., Lawrence, S. and Tsoi, A. C. (2001). Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine Learning* 44(1-2): 161-183.
- [15] Hornik, K., Stinchcombe, M. and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks* 2(5): 359-366.
- [16] Huang, W., Lai, K. K., Nakamori, Y. and Wang, S. (2004). Forecasting foreign exchange rates with artificial neural networks: A review. *International Journal of Information Technology & Decision Making* 3(1): 145-165.
- [17] Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting* 22(4): 679-688.
- [18] Jaeger, H. (2002). Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach. Technical report, Fraunhofer Institute AIS, St. Augustin-Germany.
- [19] Karunasinghe, D. S. K. and Liong, S. -Y. (2006). Chaotic time series prediction with a global model: Artificial neural network. *Journal of Hydrology* 323(1-4): 92-105.
- [20] Lapedes, A. and Farber, R. (1987). Nonlinear signal processing using neural networks: Prediction and system modelling. in: *Proceedings of the 1st IEEE International Conference on Neural Networks*, San Diego, USA.
- [21] Li, D., Han, M. and Wang, J. (2012). Chaotic time series prediction based on a novel robust echo state network. *IEEE Transactions on Neural Networks and Learning Systems* 23(5): 787-799.
- [22] Liang, F. (2005). Bayesian neural networks for nonlinear time series forecasting. *Statistics and Computing* 15(1): 13-29.
- [23] Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences* 20(2): 130-141.
- [24] Martinetz, T. M., Berkovich, S. G. and Schulten, K. J. (1993). "Neural-gas" network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks* 4(4): 558-569.
- [25] Nelson, D. B. (1991). Conditional heteroskedasticity in asset returns: A new approach. *Econometrica* 59(2): 347-370.
- [26] Ni, H. and Yin, H. (2009). A self-organising mixture autoregressive network for FX time series modelling and prediction. *Neurocomputing* 72(1618): 3529-3537.
- [27] Nicholls, D. F. and Quinn, B. G. (1982). *Random Coefficient Autoregressive Models : An Introduction*. New York, Springer-Verlog
- [28] Poli, I. and Jones, R. D. (1994). A neural net model for prediction. *Journal of the American Statistical Association* 89(425): 117-121.
- [29] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986). *Learning Internal Representations by Error Propagation*. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1, MIT Press: 318-362.
- [30] Salhi, M. S., Arous, N. and Ellouze, N. (2009). Principal temporal extensions of SOM: Overview. *International Journal of Signal Processing, Image Processing and Pattern Recognition* 2(4): 61-84.
- [31] Simon, G., Lendasse, A., Cottrell, M., Fort, J. C. and Verleysen, M. (2005). Time series forecasting: Obtaining long term trends with self-organising maps. *Pattern Recognition Letters* 26(12): 1795-1808.
- [32] Tay, F. E. H. and Cao, L. J. (2001). Application of support vector machines in financial time series forecasting. *Omega* 29(4): 309-317.
- [33] Tay, F. E. H. and Cao, L. J. (2002). Modified support vector machines in financial time series forecasting. *Neurocomputing* 48(14): 847-861.
- [34] Tsay, R. S. (1987). Conditional heteroscedastic time series models. *Journal of the American Statistical Association* 82(398): 590-604.
- [35] Tsay, R. S. (2005). *Analysis of Financial Time Series*. Hoboken, N.J. : Wiley-Interscience.
- [36] Van Gestel, T., Suykens, J. A. K., Baestaens, D. E., Lambrechts, A., Lanckriet, G., Vandaele, B., De Moor, B. and Vandewalle, J. (2001). Financial time series prediction using least squares support vector machines within the evidence framework. *IEEE Transactions on Neural Networks* 12(4): 809-821.
- [37] Varsta, M., Milln, J. D. R. and Heikkonen, J. (1997). A recurrent self-organising map for temporal sequence processing. in: *Proceedings of the 7th International Conference on Artificial Neural Networks*, Lausanne, Switzerland.
- [38] Voegtlin, T. (2002). Recursive self-organising maps. *Neural Networks* 15(8-9): 979-991.
- [39] Yao, J. and Tan, C. L. (2000). A case study on using neural networks to perform technical forecasting of forex. *Neurocomputing* 34(14): 79-98.
- [40] Yin, H. J. and Ni, H. (2009). Generalized self-organising mixture autoregressive model for modeling financial time series. in: *Proceedings of the 19th International Conference on Artificial Neural Networks*, Limassol, Cyprus.
- [41] Zhang, G., Eddy Patuwo, B. and Y. Hu, M. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting* 14(1): 35-62.