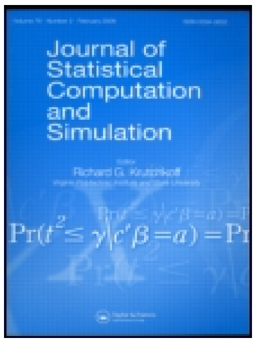

Journal of Statistical Computation and Simulation

ISSN: 0094-9655 (Print) 1563-5163 (Online) Journal homepage: http://www.tandfonline.com/loi/gscs20

Comparison of ARIMA, neural networks and hybrid models in time series: tourist arrival forecasting

Atilla Aslanargun , Mammadagha Mammadov , Berna Yazici & Senay Yolacan

To cite this article: Atilla Aslanargun , Mammadagha Mammadov , Berna Yazici & Senay Yolacan (2007) Comparison of ARIMA, neural networks and hybrid models in time series: tourist arrival forecasting, Journal of Statistical Computation and Simulation, 77:1, 29-53, DOI: 10.1080/10629360600564874

To link to this article: https://doi.org/10.1080/10629360600564874

Published online: 01 Feb 2007.

Journal of Statistical Computation and Simulation
Vol. 77, No. 1, January 2007, 29–53

Comparison of ARIMA, neural networks and hybrid models in time series: tourist arrival forecasting

ATILLA ASLANARGUN, MAMMADAGHA MAMMADOV, BERNA YAZICI and SENAY YOLACAN*

Department of Statistics, Anadolu University, 26470 Eskisehir, Turkey

(Revised 3 October 2004; in final form 7 May 2005)

For time series forecasting, different artificial neural network (ANN) and hybrid models are recommended as alternatives to commonly used autoregressive integrated moving average (ARIMA) models. Recently, combined models with both linear and nonlinear models have greater attention. In this article, ARIMA, linear ANN, multilayer perceptron (MLP), and radial basis function network (RBFN) models are considered along with various combinations of these models for forecasting tourist arrivals to Turkey. Comparison of forecasting performances shows that models with nonlinear components give a better performance.

Keywords: Time series; ARIMA; Neural networks; Backpropagation; Radial basis function network; Hybrid models

1. Introduction

As multilayer artificial neural network (ANN) models are usually nonlinear, they give more efficient results in classification, sample recognition, and forecasting problems than linear models. The statistical models used for time series analysis are usually linear. Hence, using ANN models that can impress the nonlinear structure is of great importance. Zhang *et al.* [1] presented a recent review in this area.

The most commonly used models in time series analysis are autoregressive integrated moving average (ARIMA) models. ARIMA models depend on a statistical modeling theory known as the Box–Jenkins methodology [2]. The drawback of these models is that they are linear models. The relationship between the variables is not linear for most problems in real life [3] and using linear models for such problems is not efficient. There are numerous studies comparing the performances of ANN and traditional time series techniques. For example, the empirical study results in Ansuj *et al.* [4], Caire *et al.* [5], Chin and Arthur [6], Hill *et al.* [7], Kohzadi *et al.* [8], and Maier and Dandy [9] show that the ANN give better results in forecasting accuracy.

Recently, an approach that uses ARIMA and ANN models together for time series forecasts has been recommended [10, 11]. Tseng *et al.* [10] combine the seasonal ARIMA (SARIMA)

*Corresponding author. Email: senayyolacan@anadolu.edu.tr

Journal of Statistical Computation and Simulation
ISSN 0094-9655 print/ISSN 1563-5163 online © 2007 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/10629360600564874

model with the backpropagation (BP) ANN (SARIMABP) to make better predictions. Results show that the SARIMABP model has better performance than the SARIMA model. Experimental results with real data sets [11] indicate that a hybrid methodology that combines both ARIMA and ANN models can be an effective way to improve forecasting accuracy achieved by either of the models used separately. Zhang [11] explains the reasons for using hybrid models in detail.

The use of hybrid models to improve forecasting accuracy can also be seen in papers that were published before the studies mentioned previously. Some earlier papers can be given as examples [12, 13]. To increase the forecasting accuracy in time series, using more than one model is strongly recommended for well-known time series such as M-computation [14]. Clemen [15] provides a comprehensive review and annotated bibliography in this subject area. Both theoretical and empirical findings suggest that combining different methods can be an effective and efficient way to improve forecasts [11, 16–19].

Ginzburg and Horn [20] and Pelikan *et al.* [21] proposed combining several feedforward neural networks to improve time series forecasting accuracy. Wedding and Cios [22] describe a combining methodology, using radial basis function networks (RBFN) and Box–Jenkins models. Voort *et al.* [23] used a Kohonen self-organizing map as an initial classifier, with each class having an individually tuned ARIMA model associated with it. Luxhoj *et al.* [24] presented a hybrid econometric and ANN approach for sales forecasting. Wang and Leu [25] developed a neural network trained by features extracted from ARIMA analysis. Their results show that the neural network trained by different data produced better predictions than when trained by raw data. Results from Su *et al.* [26] show that the hybrid model produced better forecasts than the ARIMA model or the neural network by itself. Hence, the combined model has produced promising results in these studies.

According to these hybrid model results, combining ANN and statistical models or different ANN models produces more accurate forecasting for time series. Choosing the hybrid model depends on the statistical and mathematical structure of the data. Of course, this relationship cannot usually be explained. As a result, the components of the hybrid models are chosen by experiment.

Some papers on tourism forecasting problems mention ANN as having better performance than statistical techniques. Unfortunately, the number of published articles that incorporate neural networks into the area of tourism-demand forecasting appears to be very limited. Law and Au [27] found that using the feedforward neural network model to forecast annual Japanese arrivals to travel to Hong Kong outperformed multiple regression, naive, moving average, and exponential smoothing in terms of forecasting accuracy. The data used in the paper in question have a linear separable structure. However, observed data can have a nonlinear structure in tourism. Law [28] extends the applicability of neural networks in tourism demand forecasting by incorporating the BP learning process into a non-linearly separable tourism demand date. The empirical results of Law [28] show that a BP neural network outperformed regression models, time series models, and feedforward neural networks in terms of forecasting accuracy. In his paper, Cho [29] investigated the applications of three time series forecasting techniques, namely exponential smoothing, univariate ARIMA, and Elman's model of ANN. Neural networks seem to be the best method for tourist arrival forecasting. The article of Kim *et al.* [30] provides a short introduction to the use of Kohonen's SOM algorithm in tourism and presents a descriptive analysis of the ANN methodology. It provides a research technique that assesses the weighting of different attributes and uses an unsupervised ANN model to describe a consumer product relationship.

In this article, the ARIMA, ANN models and hybrid models with at least one component different from ANN, and data concerning the number of monthly tourist arrivals to Turkey have been determined for time series. Their forecasting performances are examined. From this, it is seen that the model with both nonlinear components showed better performance.

2. Methodology

The ARIMA, ANN, and hybrid models used to analyse the time series covered by this study are discussed subsequently.

2.1 *ARIMA model*

Certain linear models, also known as Box–Jenkins models, are commonly used to analyse the univariate time series. In these models, any observed value of the series in any time period is defined as the linear component of the previous time t (observed values) and/or error terms. The general form of the ARIMA models is written as the following formulas [2]:

$$\text{ARIMA}(p, d, q)(P, D, Q)_s,$$

where p is the number of parameters in the autoregressive (AR) model, d the differencing degree, q the number of parameters in MA model, P the number of parameters in AR seasonal model, D the seasonal differencing degree, Q the number of parameters in MA seasonal model, and s the period of seasonality, or

$$\phi_p(B)\Phi_{Ps}(B^s)\nabla^d\nabla^{DS}X_t = \theta_q(B)\Theta_{Qs}(B^s)a_t,$$

where X_t is the observed value at time point t (or transformed value); $\phi_p(B)$ the AR operator, $[(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p)]$; B the backshift operator, $[BX_t = X_{t-1}]$; $\Phi_{Ps}(B^s)$ the seasonal AR operator, $[(1 - \Phi_1 B^s - \Phi_2 B^{2s} - \cdots - \Phi_P B^{Ps})]$; B^s the seasonal backshift operator, $[B^s X_t = X_{t-s}]$; ∇^d the differencing operator, $[\nabla^d X_t = (1 - B)^d X_t]$; ∇^{Ds} the seasonal differencing operator, $[\nabla^{Ds} X_t = (1 - B^s)^D X_t]$; a_t the random error at time point t, $[a_t \sim N(0, \sigma_a^2)]$; $\theta_q(B)$ the MA operator, $[(1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_1 B^q)]$; and $\Theta_{Qs}(B^s)$ the seasonal MA operator, $[(1 - \Theta_1 B^s - \Theta_2 B^{2s} - \cdots - \Theta_Q B^{Qs})]$.

In constructing ARIMA models, an autocorrelation function and a partial autocorrelation function are used. The distribution of the coefficients in different delays gives a differencing degree d, a seasonal differencing degree D, and the number of parameters of the AR, the moving average (MA), and seasonal AR.

2.2 *ANN approach*

2.2.1 Standard backpropagation training algorithm. Multilayer perceptrons (MLP) are used in a variety of problems, especially in forecasting. Kolmogorov's theorem of 1957 and its variations, adjusted for neural networks, show that two hidden layers are enough for a certain approximation of any complex nonlinear function. In fact, usually, one layer in a network is sufficient in order to realize a good approximation [31–33].

BP is the widespread approximation approach for training the multilayer feedforward neural networks based on the Widrow–Hoff training rule [31–32]. The main idea here is to adjust the weights and the biases that minimize the sum of the square errors by propagation of the error back at each step. In order to minimize the sum of the square errors, different BP algorithms are constructed by applying different numeric optimization algorithms among the gradient and the Newton method class.

The sum of the squares for the qth training sample in supervised training situations, with n inputs, m outputs, and p hidden units, in a two-layer (with single hidden layer) neural network is calculated as follows:

$$E^q(w) = \sum_{k=1}^{m}[y_k - t_k]^2 = \sum_{k=1}^{m}\left[f_\mathrm{o}\left(\sum_{j=1}^{p} w_{jk}^\mathrm{o} \cdot f_\mathrm{h}\left(\sum_{i=1}^{n} w_{ij}^\mathrm{h} \cdot x_i + b_j^\mathrm{h}\right) + b_k^\mathrm{o}\right) - t_k\right]^2 \tag{1}$$

where t_k is the kth target and y_k the kth output value; w_{ij}^h is the weight that connects the ith input and jth output units, w_{jk}^o the weight that connects the jth hidden and kth output units, b_j^h the bias for jth hidden unit, b_k^o the bias for kth output unit, $f_\mathrm{h}(\cdot)$ the activation function applied to the hidden units, and $f_\mathrm{o}(\cdot)$ the activation function applied to the output units. Here, w is the vector of all weight and bias components. For simplicity, the q sample indices are not shown.

A nonlinear (almost an S-shaped sigmoid) $f_\mathrm{h}(\cdot)$ function is taken. $f_\mathrm{o}(\cdot)$ can also be taken as a linear function. For all N training samples, the mean square error (MSE) is defined as:

$$E(w) = \frac{1}{n}\sum_{q=1}^{N} E^q(w). \tag{2}$$

The gradient descent numeric optimization method is used to decrease error in the standard BP training algorithm [31, 32]. The iteration of this method is as follows:

$$w_{k+1} = w_k - \alpha \cdot g_k$$

or

$$\Delta w_{k+1} = -\alpha \cdot g_k,$$

where w_k is the current vector of weights and biases, g_k the current gradient of error function (2) at the point, w_k, and α the learning (training) rate. The learning rate is crucial for BP, as it determines the magnitude of weight changes.

A standard BP training algorithm that includes *momentum* [34] is given by the improved gradient descent with momentum formula:

$$\Delta W_{k+1} = -\alpha \cdot g_k + \mu \Delta w_k,$$

where $\Delta W_k = w_k - w_{k-1}$ is the weight change in the previous iteration and μ is the momentum coefficient. The weight change in the BP training algorithm with momentum is made by the combination of the current gradient vector and the previous gradient vector. This change is better for the behavior of the algorithm, as it provides the chance for surface local minimums to escape. Of course, the better α and μ constants speed up convergence of the algorithm [35–37].

Various versions of BP training algorithms, based on the conjugate gradient (CG) and the Newton methods, which are second-order basic optimization techniques, are more efficient in classifying and forecasting problems [31, 32].

2.2.2 CG algorithms. In the CG algorithms, research is made in the conjugate directions [38]. A set of nonzero n-dimensional vectors $\{p_0, p_1, \ldots, p_{n-1}\}$ is said to be *conjugate*, with respect to the symmetric positive definite $n \times n$ matrix A if

$$p_i^{\mathrm{T}} A_{p_j} = 0 \quad \text{for all } i \neq j.$$

First, we should take into account the minimum problem of the square function:

$$F(w) = \frac{1}{2} w^{\mathrm{T}} H w - b^{\mathrm{T}} w, \tag{3}$$

where $w \in R^n$ and H is the $n \times n$ symmetric positive definite matrix. For a starting point $w_o \in R^n$, the method defined by the formulas given subsequently is called the conjugate direction method [38]:

$$w_{k+1} = w_k + \alpha_k p_k \tag{4}$$

$$\alpha_k = -\frac{p_k^{\mathrm{T}} g_k}{p_k^{\mathrm{T}} H p_k}. \tag{5}$$

Here, equation (5) is defined by the minimum problem of one variable function of $\varphi(\alpha) = F(w_k + \alpha p_k)$.

For any starting point $w_0 \in R^n$, the sequence $\{w_k\}$ generated by the conjugate direction algorithm (4) and (5) converges to the minimum point w^* of the problem (3) in at most n steps [38]. The CG method is a conjugate direction method; start by searching in the gradient direction on the first iteration

$$p_0 = -g_0. \tag{6}$$

The conjugate current p_k vector is calculated by using the previous p_{k-1} vector and the current gradient

$$p_k = -g_k + \beta_k p_{k-1}. \tag{7}$$

The coefficient β_k in equation (7) is selected by the condition of the p_{k-1} and p_k vectors, being conjugate with respect to the symmetric positive definite $n \times n$ matrix $H(p_{k-1} H p_k = 0)$:

$$\beta_k = \frac{g_k^{\mathrm{T}} H p_{k-1}}{p_{k-1}^{\mathrm{T}} H p_{k-1}}. \tag{8}$$

There are various CG algorithms dependent on different calculation formulas of the β_k coefficients. The important ones are Bishop [31], Haykin [32], and Nocedal and Wright [38].

The Fletcher–Reeves CG algorithm β_k is defined by

$$\beta_k = \frac{g_k^{\mathrm{T}} g_k}{g_{k-1}^{\mathrm{T}} g_k}.$$

This is the ratio of the norm squared of the current gradient to the norm squared of the previous gradient. Another version of the CG algorithm was proposed by Polak and Ribere, for which β_k is defined by

$$\beta_k = \frac{g_k^{\mathrm{T}} (g_k - g_{k-1})}{g_{k-1}^{\mathrm{T}} g_{k-1}}.$$

Now, error function (2) is taken into account. As activation functions are nonlinear, the sum of the mean squared error function $E(w)$ is the nonlinear function of w. Using the first two terms of the Taylor series expansion of $E(w)$ around w_k, we can approximate the function $E(w)$ to a squared function

$$E(w_{k+1}) - E(w_k) = \nabla E(w_{k+1}) \approx g_k^{\mathrm{T}} \Delta w_k + \frac{1}{2} \Delta w_k^{\mathrm{T}} H_k \Delta w_k, \tag{9}$$

where H_k is the Hessian matrix in current point w_k:

$$H_k = \frac{\partial^2 E(w_k)}{\partial w^2}.$$

Hence, in each current step, taking function (9) instead of function (3), a suitable minimization problem is taken into account and a local CG is realized by using equations (4)–(8). In this case, the Hessian matrix H_k is assumed as positive definite.

Another CG algorithm is the scaled CG (SCG) algorithm. The CG algorithms reviewed earlier to make a line search at each iteration makes calculations very time consuming.

Moller [39] proposed a SCG algorithm that does not use a line search procedure in traditional CG algorithms. The basic idea of an SCG is to combine the model trust region approach with the CG approach [31, 38].

Moller proposed an Hp_k evaluation with O(W) order operations (W being the total number of weights and biases) instead of using line minimization, which involves few error function evaluations. This approach may prove unsuccessful when the Hessian matrix H in local approximation is not positive definite. In this case, the denominator of equation (5) $p_k^{\mathrm{T}} H p_k$ will be negative, and finally, the weight update can increase the value of error function. The problem can be overcome by modifying the Hessian matrix to ensure that it is positive definite. For example,

$$H_{\mathrm{mod}} = H + \lambda I,$$

where I is the unit matrix and $\lambda \geq 0$ is a scaling coefficient. We can make the Hessian matrix positive definite by choosing the large positive value of λ. The step length formula (5) is changed as follows:

$$\alpha_k = -\frac{p_k^{\mathrm{T}} g_k}{p_k^{\mathrm{T}} H p_k + \lambda_k \| p_k \|^2}. \tag{10}$$

α_k decreases while λ_k increases. The size of the trust region is adjusted by the λ_k parameter. As λ_k increases, the trust region gets smaller. The choice of an appropriate λ_k is very important for each step of the SCG algorithm. For example, Moller [39] chooses an appropriate λ_k for each step, transforming the denominator of equation (10) to $-p_k^{\mathrm{T}} H_k p_k > 0$.

2.2.3 Quasi-Newton algorithms. The basic step of Newton's method is

$$w_{k+1} = w_k - H_k^{-1} g_k, \tag{11}$$

where H_k is the Hessian matrix in the current point w_k. Newton's method often converges faster than CG methods. Unfortunately, it is complex and expensive to compute the Hessian matrix. Quasi-Newton (QN) (or secant) methods are based on Newton's method, but do not require calculation of second derivatives. They update (the update is computed as a function of the gradient) an approximate Hessian matrix at each iteration of the algorithm [38].

The Broyden, Fletcher, Goldfarb, and Shanno (BFGS) algorithm is a successful QN algorithm. In the BFGS algorithm, instead of the inverse of Hessian matrix in equation (11), its G approach is constructed in the following recursive manner:

$$G_{k+1} = G_k + \frac{pp^{\mathrm{T}}}{p^{\mathrm{T}}v} - \frac{G_k vv^{\mathrm{T}} G_k}{v^{\mathrm{T}} G_k v} + (v^{\mathrm{T}} G_k v)uu^{\mathrm{T}},$$

where

$$p = w_{k+1} - w_k, \quad v = g_{k+1} - g_k, \quad u = \frac{p}{p^{\mathrm{T}}v} - \frac{G_k v}{v^{\mathrm{T}} G_k v}.$$

As the G matrix is positive definite, $-Gg$ will be the decreasing direction of the algorithm. A weight update is made as follows:

$$w_{k+1} = w_k + \alpha_k G_k g_k,$$

where α_k is found by line minimization.

The one-step secant algorithm is an attempt to bridge the gap between the CG algorithms and QN algorithms. This algorithm accepts that the previous Hessian is the identity matrix, and hence, inverse matrixes are not calculated to determine the new search direction. This algorithm requires very light and less storing at each step than CG algorithms.

The Levenberg–Marquart (LM) algorithm is one of the required QN algorithms [31]. Hessian matrixes are not calculated and, being of second order, were not taken into account. The Hessian matrix can be approximated as $H = J^{\mathrm{T}}J$, where J is the Jacobian matrix that contains the first derivatives of the network errors with respect to the weights and biases. The gradient can be computed as $g = J^{\mathrm{T}}\varepsilon$, where ε is a vector of network error. The LM algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$w_{k+1} = w_k + [J^{\mathrm{T}}J + \mu I]^{-1} J^{\mathrm{T}}\varepsilon.$$

This algorithm runs fast for moderate-dimensioned feedforward neural networks for regression problems.

2.2.4 Radial basis function networks. RBFN [31, 32] are also used besides MLP networks in regression and classification problems. In RBFN, one hidden layer with the required number of units is enough in order to model a function. The activations of hidden (radial) units are defined depending on the distance of the input vector and the center vector. Typically, the radial layer has exponential activation functions and the output layer a linear activation function. Appropriate y output vector for the x input vector is calculated as follows for n input, m output, and p radial units for RBFN:

$$y_k(x) = \sum_{j=0}^{p} w_{kj}\phi_j(x), \quad k = 1, 2, \ldots, m, \tag{12}$$

where w_{kj}, $j = 1, 2, \ldots, p$, are the appropriate weights for kth output unit, $\phi_j(x)$, $j = 1, 2, \ldots, p$, is the basis function of jth radial unit, w_{k0}, $k = 1, 2, \ldots, m$, are the appropriate deviations for kth output unit, and ϕ_0 is an extra basis function with the activation value fixed at $\phi_0 = 1$. Usually, more attention is paid to the following Gaussian basis function:

$$\phi_j(x) = \exp\left(\frac{-\|x - \mu_j\|^2}{2\sigma_j}\right), \quad j = 1, 2, \ldots, p \tag{13}$$

where $\mu_j = (\mu_{j1}, \ldots, \mu_{jn})$ vector is the center for $\phi_j(x)$ and σ_j is the deviation (or width) parameters of that function. The basis function of the unit is defined using those two parameters. Equation (12) can be written in matrix notation as:

$$y(x) = W \cdot \phi, \tag{14}$$

where $W = (w_{kj})$ and $\phi = (\phi_j)$. As can be seen in equation (12), the linear activation function is used in RBFN for the output layer.

Education is made in three stages in RBFN. In the first stage, by unsupervised education, radial basis function centers (in other words μ_j) are optimized using all $\{x^{(i)}\}$, $i = 1, 2, \ldots, N$, education data. Centers can be assigned by a number of algorithms: subsampling, K-means, Kohenen training, or learned vector quantization. In the second stage, σ_j, $j = 1, 2, \ldots, p$, parameters can be assigned by algorithms explicit, isotropic and K-nearest neighbor. In the third stage of education, the basis functions that are obtained for adjusting the appropriate weights for output units are taken as fixed and deviation parameters are added to the linear sum. Optimum weights are obtained by minimization of the sum of the square errors,

$$E(w) = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{m}[y_k(x^{(i)}) - t_k^{(i)}]. \tag{15}$$

In equation (15), $t_k^{(i)}$ is the target value for output unit k when the network is presented with input vector $x^{(i)}$, $i = 1, 2, \ldots, N$. As function (14) is the quadratic function of the weights, optimum weights can be found as the solution of the linear equations system. The output layer is usually optimized using the pseudo-inverse technique.

MLP, with a defined architecture, is given by the appropriate weights and the biases of the units, but in RBFN, it is given by the center and the deviation of the radial units and by the weights and biases of the output units n. As the point is given by n coordinates in n-dimensional space, the number of the coordinates is equal to the linear input units n. Hence, in ST Neural Network software, the coordinates of the center radial unit are taken as weights and the deviation of the radial unit is taken as bias. As a result, radial weights denote the center point and radial bias denotes the deviation.

Having only one hidden layer and making faster education than MLP are the advantages of RBFN. As the linear modeling methods are more useful in output layers of RBFN, the difficulties that occur about the local minimums in MLP are removed.

However, RBFN has some disadvantages compared with MLP. In order to correctly model a typical function in RBFN, many more hidden (radial) units may be required than appropriate in the MLP model. That may cause the model to slow down, and more memory may be required. RBFN is very sensitive to any increment of the network dimension and some difficulties can occur as a result of an increment in the number of input units.

RBFN is unsuccessful in extrapolation in its nature. MLP networks are more successful in extrapolation problems than RBFN because when the input data are far from the radial centers, the output signal is 0, and this may not show the required result.

2.2.5 Linear neural networks. In ANNs, when a linear model is mentioned, a network is usually thought to be without a hidden layer and with the linear unit in the output layer. In ST Neural Network software, the linear network is educated by a standard linear optimization algorithm depending on the pseudo-inverse technique. The pseudo-inverse technique is fast and guaranteed to find the optimal solution. This is the best way to solve a very complicated problem, using a very simple method. In general, when the number of education data is not large, a complicated model is not required.

2.3 *ANN approach to time series modeling*

The feedforward ANN is commonly used for time series modeling and forecasting [33]. For one hidden layer network architecture $n{:}p{:}1$ (n, number of inputs; p, number of hidden units; and 1, number of outputs), the inputs are the observed values of nth previous time points and the outputs (targets) are the $(n + 1)$th observed values. Examining equations (1) and (2), it can be seen that ANN are nonlinear functions of previous observations $(y_{t-1}, y_{t-2}, \dots, y_{t-n})$ to future observations (y_t) [11]:

$$y_t = f(y_{t-1}, y_{t-2}, \dots, y_{t-n}, w) + \varepsilon_t,$$

where $(y_{t-1}, y_{t-2}, \dots, y_{t-n})$ are the input values, y_t the target value, w the weights of the network, and ε_t the vector of biases at time point t. The $\hat{y}_t$ prediction value is calculated as follows:

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, \dots, y_{t-n}, w).$$

If N number of $(y_1, y_2, \dots, y_N)$ observations are used for a time series and a one step forward forecast is made, the number of training samples are $N - n$. $(y_1, y_2, \dots, y_n)$ is taken as a first input training sample and y_{n+1} is accepted as the target. The second training pattern will contain $(y_2, y_3, \dots, y_{n+1})$ as inputs and y_{n+2} as the second target output. Finally, $(y_{N-n}, y_{N-n+1}, \dots, y_{N-1})$ and y_N will be the last input pattern and target, respectively.

In training procedure, with the help of different BP algorithms, the parameters (weights and biases) of the network are obtained by getting closer to the minimum value of the sum of the square errors.

$$\mathrm{SSE} = \sum_{t=n+1}^{N} (y_t - \hat{y}_t)^2.$$

2.4 *Hybrid methodology*

There have been numerous papers that mention that hybrid models show better performance than the use of one linear model or one nonlinear model for forecasting in time series. Nevertheless, different ideas have been proposed concerning the components of hybrid models. In recent papers [10, 11], more importance is given to hybrid models that are composite of ARIMA and NN models. In our opinion, the proposed hybrid model does not always show a better performance. Various combining methods have been proposed in the literature [20–22, 24]. Pelikan *et al.* [21] and Ginzburg and Horn [20] propose combining several feedforward neural networks to improve time series forecasting accuracy. Granger [40] has pointed out that for a hybrid model to produce superior forecasts, the component model should be suboptimal. In general, it has been observed that it is more effective to combine individual forecasts that are based on different information sets [11, 40, 41]. The basic idea of model combination in forecasting is to use each model's unique feature to capture different patterns in the data [11].

For time series sampling in this study, ARIMA&NN and different NN&NN hybrid models are evaluated with experimental calculations, with interesting results obtained. As a result of the experiment, it is seen that hybrid models like MLP&MLP and MLP&RBFN with two nonlinear components show better performance in forecasting problems. Hence, the hybrid model structure of Zhang [11] can be extended. This means that both of the components of the hybrid model may be nonlinear at the same time:

$$y_t = y_t^1 + y_t^2.$$

In this model, y_t is the observation value at time point t, y_t^1 and y_t^2 are the linear or nonlinear model components, and superscripts denote the row number of the model. First, the model indiced 1 is applied to the observation data and $e_t^1 = y_t - \hat{y}_t^1$, then the others are calculated. Here, $\hat{y}_t^1$ is the forecast value of the first model at time point t. If the first model contains m_1 input units, the number of e_t^1 units will be $N - m_1$. If the second model contains m_2 input units, the number of $\hat{y}_t^2$ forecast values will be $N - m_1 - m_2$. In this case, the forecast values appropriate for the second model are calculated as follows:

$$\hat{y}_t^2 = f_2(e_{t-1}, e_{t-2}, \ldots, e_{t-m_2}),$$

where f_2 is the function obtained from the second model. The forecast for the combined model is defined as follows:

$$\hat{y}_t = \hat{y}_t^1 + \hat{y}_t^2.$$

The adjusted forecasts are calculated as the sums of the first and second models. The hybrid model with good performance is obtained by the evaluation measure for the forecasting.

2.5 *Forecast evaluation methods*

The performance of the model is related with how close the forecast values are for test data and the observed values. Three different forecast consistency measures are used in order to compare the performances of obtained ARIMA, NN, and different hybrid models: MSE (or root MSE (RMSE)), mean absolute error (MAE), and mean absolute percentage error (MAPE), respectively. These measures are defined by taking y_t as the observation value at time point t, $\hat{y}_t$ as the forecast value at time point t, and n as the number of forecast as follows:

1. MSE or RMSE:

$$\text{MSE} = \frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2 \text{ or RMSE } = \sqrt{\text{MSE}}.$$

2. MAE:

$$\text{MAE} = \frac{1}{T}\sum_{t=1}^{T}|y_t - \hat{y}_t|.$$

3. MAPE:

$$\text{MAPE} = \frac{1}{T}\sum_{t=1}^{T}\frac{|y_t - \hat{y}_t|}{|y_t|}(100\%).$$

MSE (RMSE) and MAE measures are absolute performance measures, whereas MAPE is a relative performance measure [42]. Although there have been numerous discussions on comparison of these forecasting models, the most common measure is MSE [33, 43, 44].

3. Experimental evaluation

In this section, the time series of the number of monthly tourist arrivals to Turkey is examined. Appropriate ARIMA, ANN, and hybrid models were chosen by experiments in order to make forecasts, and these models are also compared.

In this study, STATISTICA Neural Networks and SPSS statistical packages are used.

3.1 *Data set*

The data set, the number of monthly tourist arrivals to Turkey between January 1984 and December 2003, is taken from the Prime Ministry State Institute of Statistics [45], Turkey.

The data set is divided into two parts for the use in training and forecasting. In the first part, 216 months' data are taken into account for the January 1984–December 2001 period. These data are used in training to construct the models. In the second part, with the help of the models constructed in the first part, the performances of those models are calculated using the 24 months' data for the January 2002–December 2003 period.

3.2 *Constructing the appropriate ARIMA model*

The graph of 216-month period for January 1984–December 2001 is given in figure 1. Examining figure 1, it is seen that the series is not stable in variance after the logarithmic transformation variance stability criterion is provided, and it can also be seen that the series includes seasonal and trend effects. Hence, it was decided that the most appropriate model for this series is ARIMA$(1, 1, 1)(1, 1, 0)_{12}$. This model can be written in open form as follows:

$$(1 - 0.74565B)(1 + 0.31841B^{12})(1 - B)(1 - B^{12})\log(X_t) = (1 - 0.9882B)a_t.$$

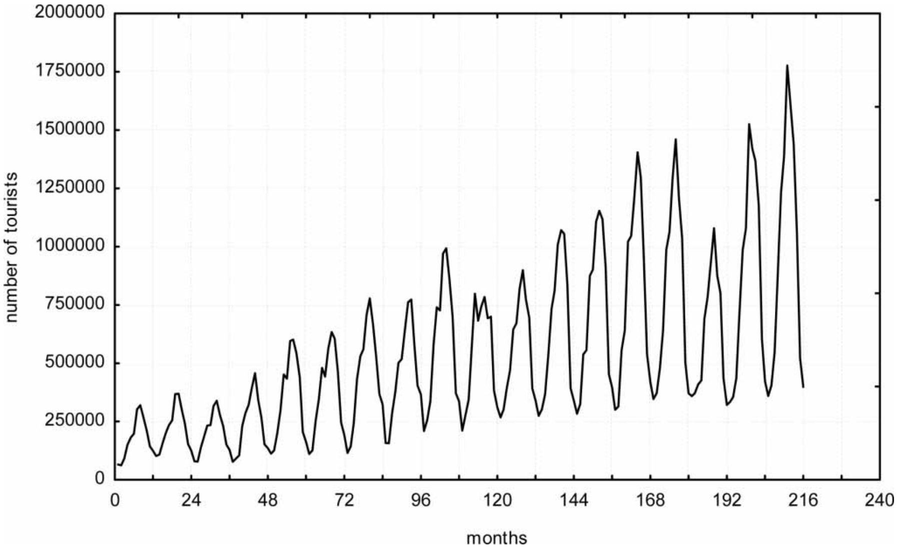

Figure 1. Number of monthly tourist arrivals to Turkey (1984–2001).

3.3 *Choice of appropriate ANN models*

The 216 months' data of tourist arrivals to Turkey for January 1984–December 2001 period were used in training of the network. An evaluation of the model was made depending on the forecasts for the 24-month period between January 2002 and December 2003. The choice of the best model depends on a comparison of statistics such as the MSE (RMSE), MAE, and MAPE. As the initial weight and bias values of the network were random, 150 replications were made for the same network structure, and the models giving the best forecasts were determined.

As the tourist arrival data in question included seasonality, after trying many neural networks with different numbers of input units, as expected, the number of input units was determined as 12. During these experiments, various neural network algorithms with single layer, with one or two hidden layers, multilayer feedforward algorithms, and RBFN models were applied on the sample data set. As the initial 12 data were lost because of the seasonality, 204 from the 216 data were used to adjust the weights. In the training stage of the network, data were divided into two parts: 132 of the 204 data were used for training and 72 data were used for validation. This division was used to restrict memorization of the network and provided for better forecasts [31, 32].

The single layer neural network with (12:1) architecture (12 input units, one output unit, and one bias) showed better performance for the forecast data as a result of the adjustment of the suitable weights in training. The linear network is trained by using the pseudo-inverse method [31]. The biases and the weights obtained by training are given in table 1.

The neural network that showed the best performance among the MLP for the time series in question was found to be the (12:1:1) MLP. The QN method showed the best performance in 33 epochs. The hyperbolic tanjant function is applied in the hidden layer and the linear activation function is applied in the output unit. The biases and weights resulting from training are given in table 2.

Among the RBF networks, the (12:48:1) RBFN showed the best performance. In the radial units exponential and in the output units, linear activation functions are used. The weights and the deviations are given in table 3.

3.4 *Constructing the appropriate hybrid models*

In this study, hybrid models such as the ARIMA&MLP, LinearNN&MLP, ARIMA&RBFN, LinearNN&RBFN and MLP&MLP, MLP&RBFN, RBFN&MLP, and RBFN&RBFN were

Table 1. (12:1)LinearNN biases and weights.

Tresh	−0.0117
1.1	0.23077
1.2	0.29894
1.3	−0.1119
1.4	0.05746
1.5	−0.1224
1.6	0.40528
1.7	−0.519
1.8	0.46057
1.9	−0.3332
1.10	−0.0895
1.11	0.01515
1.12	0.71798

Table 2. (12:1:1) Biases and weights of MLP.

	2.1	3.1
Tresh	−0.553994	−0.62866
1.1	−0.1244	
1.2	−0.20695	
1.3	0.070974	
1.4	−0.227015	
1.5	0.20911	
1.6	−0.324741	
1.7	0.437578	
1.8	−0.426368	
1.9	0.398016	
1.10	−0.066949	
1.11	0.007901	
1.12	−0.662026	
2.1		−1.19608

Note: The row and column header numeric terminology first lists the layer, then the unit number within the layer. For example, 2.1 stands for unit 1 in layer 2.

taken into account. In determining the ARIMA&MLP and the ARIMA&RBFN hybrid models, first, the ARIMA(1,1,1)(1,1,0)$_{12}$ model was applied to the data. A series of residuals, the differences between the observed values and the estimates of these models, was obtained. In the series of residuals, as a first-order difference and a first-order seasonal differences were taken in the ARIMA model, 13 data were lost and 203 data remained. At the next step, the MLP or RBFN models were applied to the series of residuals. According to the forecasts resulting from experiment, the hybrid model with the best performance among the ARIMA&MLP hybrid models is ARIMA(1,1,1)(1,1,0)$_{12}$&(6:8:1)MLP. For training, 139 data were used, and 58 data were used for validation when the MLP model ran on residual values. The BP algorithm with momentum among the MLP algorithms showed the best performance at the second epoch (BP2b). A hyperbolic tanjant was applied in the hidden layers and a linear activation function was applied in the output unit. The weights and the biases of the appropriate neural network are given in table 4.

The ARIMA(1,1,1)(1,1,0)$_{12}$&(6:7:1)RBFN among the ARIMA&RBFN hybrid models showed the best performance. The weights and the biases of the (6:7:1)RBFN are given in table 5.

In the process of determining the LinearNN&MLP hybrid model, (12:1) Linear NN&(6:8:1)MLP hybrid model showed the best performance. First, a linear neural network was applied to the data and then the MLP network was applied. For training, 140 data were used, and 58 data were used for validation when the MLP model was used on residuals. Here again, the BP algorithm showed the best performance in the 21st epoch. The weights and biases of the appropriate MLP model are shown in table 6.

Similarly, the (12:1)LinearNN&(9:15:1)RBFN was chosen among the LinearNN&RBFN hybrid networks. During construction of the (9:15:1) RBFN component, 138 of the deviations were used for education and 57 of them were used for validation. The weights and the deviations of the appropriate RBFN are given in table 7.

Among the MLP&MLP hybrid models, the (12:1:1)MLP&(5:1:1:1)MLP showed the best performance. Here, the (12:1:1)MLP neural network was the same as the one determined in section 3.3. A hyperbolic tanjant was used in the hidden layers in that network and a linear activation function was used in the output unit. The CG algorithm showed the best performance at the 26th epoch. The weights and the biases of the (12:1:1)MLP&(5:1:1:1)MLP network are shown in table 8 as the results of training.

Table 3. Biases and weights of (12:48:1)RBFN.

	2.1	2.2	2.3	2.4	2.5	2.6	2.7
Thresh	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1.1	0.117960	0.110560	0.157626	0.090502	0.030041	0.220587	0.038469
1.2	0.090502	0.044000	0.236650	0.043431	0.000000	0.187379	0.027944
1.3	0.043431	0.034290	0.215142	0.030041	0.007571	0.131463	0.014597
1.4	0.030041	0.020392	0.285254	0.000000	0.016067	0.139368	0.018030
1.5	0.000000	0.028619	0.327284	0.007571	0.089948	0.281127	0.046576
1.6	0.007571	0.072456	0.311013	0.016067	0.122726	0.330919	0.072081
1.7	0.016067	0.130360	0.226799	0.089948	0.143663	0.555167	0.092481
1.8	0.089948	0.220293	0.099078	0.122726	0.186086	0.569933	0.104536
1.9	0.122726	0.209816	0.067490	0.143663	0.223252	0.664227	0.171056
1.10	0.143663	0.304773	0.022118	0.186086	0.155150	0.781125	0.171721
1.11	0.186086	0.308273	0.038688	0.223252	0.110560	0.717912	0.131071
1.12	0.223252	0.274448	0.097400	0.155150	0.044000	0.511955	0.096988

	2.8	2.9	2.10	2.11	2.12	2.13	2.14
Thresh	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1.1	0.489555	0.142666	0.072456	0.584202	0.333726	0.186086	0.175968
1.2	0.220587	0.111486	0.130360	0.575161	0.349464	0.223252	0.249836
1.3	0.187379	0.078711	0.220293	0.446379	0.437176	0.155150	0.259107
1.4	0.131463	0.038469	0.209816	0.185690	0.483548	0.110560	0.328575
1.5	0.139368	0.027944	0.304773	0.154763	0.409853	0.044000	0.402667
1.6	0.281127	0.014597	0.308273	0.121308	0.363668	0.034290	0.409561
1.7	0.330919	0.018030	0.274448	0.145608	0.185111	0.020392	0.296525
1.8	0.555167	0.046576	0.213409	0.270681	0.155388	0.028619	0.192306
1.9	0.569933	0.072081	0.075200	0.281659	0.116050	0.072456	0.169829
1.10	0.664227	0.092481	0.050383	0.469280	0.132366	0.130360	0.077247
1.11	0.781125	0.104536	0.019621	0.485211	0.171080	0.220293	0.104492
1.12	0.717912	0.171056	0.028580	0.605391	0.269510	0.209816	0.153065

	2.15	2.16	2.17	2.18	2.19	2.20	2.21
Thresh	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1.1	0.425482	0.152176	0.038688	0.633350	0.140806	0.361222	0.136728
1.2	0.210857	0.078914	0.097400	0.611974	0.153854	0.416142	0.111917
1.3	0.143194	0.115453	0.207228	0.489555	0.117960	0.502795	0.132015
1.4	0.150907	0.156712	0.266032	0.220587	0.090502	0.589507	0.186328
1.5	0.163013	0.294713	0.283517	0.187379	0.043431	0.470056	0.230134
1.6	0.210485	0.423789	0.370938	0.131463	0.030041	0.425482	0.333726
1.7	0.378767	0.355977	0.412112	0.139368	0.000000	0.210857	0.349464
1.8	0.534855	0.392119	0.347126	0.281127	0.007571	0.143194	0.437176
1.9	0.589447	0.416077	0.264009	0.330919	0.016067	0.150907	0.483548
1.10	0.852236	0.362195	0.170553	0.555167	0.089948	0.163013	0.409853
1.11	0.789580	0.365989	0.144611	0.569933	0.122726	0.210485	0.363668
1.12	0.759742	0.179636	0.047196	0.664227	0.143663	0.378767	0.185111

	2.22	2.23	2.24	2.25	2.26	2.27	2.28
Thresh	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1.1	0.611974	0.007571	0.102432	0.062115	0.210857	0.061419	0.485211
1.2	0.489555	0.016067	0.157626	0.091210	0.143194	0.070423	0.605391
1.3	0.220587	0.089948	0.236650	0.092506	0.150907	0.132550	0.633350
1.4	0.187379	0.122726	0.215142	0.140806	0.163013	0.142666	0.611974
1.5	0.131463	0.143663	0.285254	0.153854	0.210485	0.111486	0.489555
1.6	0.139368	0.186086	0.327284	0.117960	0.378767	0.078711	0.220587
1.7	0.281127	0.223252	0.311013	0.090502	0.534855	0.038469	0.187379
1.8	0.330919	0.155150	0.226799	0.043431	0.589447	0.027944	0.131463
1.9	0.555167	0.110560	0.099078	0.030041	0.852236	0.014597	0.139368
1.10	0.569933	0.044000	0.067490	0.000000	0.789580	0.018030	0.281127
1.11	0.664227	0.034290	0.022118	0.007571	0.759742	0.046576	0.330919
1.12	0.781125	0.020392	0.038688	0.016067	0.647901	0.072081	0.555167

(*continued*)

Table 3. Continued.

	2.29	2.30	2.31	2.32	2.33	2.34	2.35
Thresh	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1.1	0.091210	0.050383	0.044000	0.789580	0.155388	0.525450	0.402667
1.2	0.092506	0.019621	0.034290	0.759742	0.116050	0.538541	0.409561
1.3	0.140806	0.028580	0.020392	0.647901	0.132366	0.461173	0.296525
1.4	0.153854	0.102432	0.028619	0.308898	0.171080	0.364946	0.192306
1.5	0.117960	0.157626	0.072456	0.203662	0.269510	0.172658	0.169829
1.6	0.090502	0.236650	0.130360	0.165857	0.385396	0.152176	0.077247
1.7	0.043431	0.215142	0.220293	0.192534	0.431311	0.078914	0.104492
1.8	0.030041	0.285254	0.209816	0.276514	0.547997	0.115453	0.153065
1.9	0.000000	0.327284	0.304773	0.475084	0.584202	0.156712	0.295143
1.10	0.007571	0.311013	0.308273	0.679137	0.575161	0.294713	0.389824
1.11	0.016067	0.226799	0.274448	0.771168	0.446379	0.423789	0.382223
1.12	0.089948	0.099078	0.213409	1.000000	0.185690	0.355977	0.525450

	2.36	2.37	2.38	2.39	2.40	2.41	2.42
Thresh	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1.1	0.328575	0.018030	0.028580	0.852236	0.132366	0.294713	0.264009
1.2	0.402667	0.046576	0.102432	0.789580	0.171080	0.423789	0.170553
1.3	0.409561	0.072081	0.157626	0.759742	0.269510	0.355977	0.144611
1.4	0.296525	0.092481	0.236650	0.647901	0.385396	0.392119	0.047196
1.5	0.192306	0.104536	0.215142	0.308898	0.431311	0.416077	0.046872
1.6	0.169829	0.171056	0.285254	0.203662	0.547997	0.362195	0.121557
1.7	0.077247	0.171721	0.327284	0.165857	0.584202	0.365989	0.175968
1.8	0.104492	0.131071	0.311013	0.192534	0.575161	0.179636	0.249836
1.9	0.153065	0.096988	0.226799	0.276514	0.446379	0.136728	0.259107
1.10	0.295143	0.044863	0.099078	0.475084	0.185690	0.111917	0.328575
1.11	0.389824	0.027715	0.067490	0.679137	0.154763	0.132015	0.402667
1.12	0.382223	0.001195	0.022118	0.771168	0.121308	0.186328	0.409561

	2.43	2.44	2.45	2.46	2.47	2.48	3.1
Thresh	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	−0.000001
1.1	0.044863	0.234963	0.355977	0.170553	0.131071	0.269510	
1.2	0.027715	0.332398	0.392119	0.144611	0.096988	0.385396	
1.3	0.001195	0.534773	0.416077	0.047196	0.044863	0.431311	
1.4	0.000290	0.579922	0.362195	0.046872	0.027715	0.547997	
1.5	0.036086	0.712607	0.365989	0.121557	0.001195	0.584202	
1.6	0.062115	0.813608	0.179636	0.175968	0.000290	0.575161	
1.7	0.091210	0.666011	0.136728	0.249836	0.036086	0.446379	
1.8	0.092506	0.563607	0.111917	0.259107	0.062115	0.185690	
1.9	0.140806	0.250194	0.132015	0.328575	0.091210	0.154763	
1.10	0.153854	0.172764	0.186328	0.402667	0.092506	0.121308	
1.11	0.117960	0.165696	0.230134	0.409561	0.140806	0.145608	
1.12	0.090502	0.173158	0.333726	0.296525	0.153854	0.270681	

	3.1		3.1		3.1
2.1	0.000144	2.17	−0.000056	2.33	0.000005
2.2	0.000040	2.18	−0.000041	2.34	−0.000050
2.3	−0.000065	2.19	−0.000039	2.35	−0.000052
2.4	0.000011	2.20	0.000067	2.36	0.000045
2.5	0.000061	2.21	0.000013	2.37	−0.000160
2.6	−0.000014	2.22	0.000015	2.38	0.000004
2.7	−0.000167	2.23	0.000091	2.39	0.000021
2.8	−0.000032	2.24	0.000291	2.40	−0.000001
2.9	−0.000060	2.25	0.000194	2.41	−0.000038
2.10	−0.000261	2.26	−0.000001	2.42	−0.000034
2.11	0.000075	2.27	−0.000108	2.43	0.000129
2.12	−0.000058	2.28	−0.000021	2.44	−0.000003
2.13	0.000095	2.29	0.000033	2.45	0.000022
2.14	0.000059	2.30	0.000010	2.46	0.000000
2.15	0.000038	2.31	−0.000057	2.47	−0.000142
2.16	0.000009	2.32	−0.000007	2.48	−0.000005

Table 4. Weights and biases of the (6:8:1)MLP in ARIMA(1,1,1)(1,1,0)$_{12}$&(6:8:1)MLP hybrid model.

	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	3.1
Thresh	0.79566	0.55945	−0.91544	−0.07465	0.15788	0.40383	−0.98710	0.38410	−0.50855
1.1	0.56391	−0.01913	−0.50720	−0.70253	0.47862	0.89161	0.42540	0.75215	
1.2	−0.40523	−0.61191	0.45859	0.39600	−0.74939	−0.42013	−0.31570	−0.72924	
1.3	0.50098	0.63810	0.97018	0.05238	0.43135	−0.25028	−0.31026	−0.07264	
1.4	0.09976	−0.82663	0.50246	−0.80849	1.05009	0.27794	0.09698	−0.31619	
1.5	−0.34296	0.67088	0.16878	0.21697	0.50562	0.23690	−0.40151	−0.23271	
1.6	−0.27098	−0.70050	−0.11548	0.30347	0.40983	0.20608	1.02916	−0.20098	
2.1									−0.50233
2.2									0.30308
2.3									0.26610
2.4									−0.17124
2.5									−0.44225
2.6									0.88190
2.7									−0.54051
2.8									−0.52367

Table 5. Weights and biases of the (6:7:1)RBFN in ARIMA(1,1,1)(1,1,0)$_{12}$&(6:7:1)RBFN hybrid model.

	2.1	2.2	2.3	2.4	2.5	2.6	2.7	3.1
Thresh	3.293101	1.786022	2.509460	4.093233	2.276624	4.658082	2.462029	−0.000018
1.1	0.623399	0.300301	0.530617	0.410070	0.570332	0.372532	0.393418	
1.2	0.418619	0.594181	0.377263	0.428830	0.146319	0.414357	0.450726	
1.3	0.362787	0.442617	0.565808	0.409004	0.576980	0.433841	0.125704	
1.4	0.491916	0.331511	0.453753	0.504908	0.393989	0.362318	0.486754	
1.5	0.330896	0.663406	0.226098	0.437954	0.470239	0.405638	0.377788	
1.6	0.307160	0.237956	0.572709	0.539600	0.422514	0.377649	0.394136	
2.1								0.000028
2.2								−0.000015
2.3								−0.000050
2.4								0.000033
2.5								−0.000005
2.6								0.000003
2.7								−0.000021

Table 6. Weights and biases of the (6:8:1)MLP in (12:1)LinearNN&(6:8:1)MLP hybrid model.

	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	3.1
Thresh	−0.80817	0.63424	0.06267	0.70848	−0.09402	0.70937	1.01132	−0.51214	−0.37417
1.1	0.46057	0.53273	−0.07746	0.40246	0.19174	−0.45590	−0.30790	−1.14909	
1.2	0.52244	0.90867	−0.86241	−0.05747	0.17881	0.37828	−0.03612	0.72575	
1.3	0.16787	0.76981	0.39791	−0.98597	0.35686	0.23664	−0.11163	0.41984	
1.4	0.24685	0.43558	−0.22104	−0.77655	0.76505	−0.60545	0.16487	0.29778	
1.5	0.81519	1.08550	−0.81870	0.85917	0.98391	−0.66661	0.58449	−0.57349	
1.6	−0.54402	0.13778	−0.96081	−0.30198	0.04617	−0.89985	−0.91242	0.86253	
2.1									−0.84903
2.2									−0.20860
2.3									−0.66474
2.4									−0.46100
2.5									0.60488
2.6									−0.14983
2.7									0.29942
2.8									−0.34688

Table 7. Weights and biases of (9:15:1)RBF in (9:15:1)RBF & (12:1)LinearNN hybrid model.

	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8
Thresh	2.891948	1.197365	2.061988	4.262847	2.720863	1.762771	2.093812	0.963355
1.1	0.552145	0.542420	0.281831	0.439961	0.598050	0.496203	0.874885	0.464050
1.2	0.661837	0.854248	0.429251	0.508052	0.602225	0.403749	0.602524	0.459921
1.3	0.432656	0.612911	0.253369	0.554884	0.776487	0.333885	0.631747	0.392052
1.4	0.608131	0.535927	0.427229	0.618742	0.438268	0.409895	0.708576	0.300749
1.5	0.454445	0.947982	0.376245	0.430149	0.530190	0.494920	0.590313	0.187518
1.6	0.573946	0.570433	0.556891	0.578994	0.718518	0.143535	0.488141	0.223540
1.7	0.583269	0.528202	0.376083	0.433484	0.602051	0.347840	0.657451	0.171198
1.8	0.680736	0.860004	0.514208	0.583004	0.450735	0.359088	0.406257	0.190136
1.9	0.389645	0.532320	0.401962	0.575996	0.626137	0.364750	0.566248	0.215411

	2.9	2.10	2.11	2.12	2.13	2.14	2.15	3.1
Thresh	4.704492	1.862662	1.214924	2.099405	4.510473	4.025605	4.155095	−0.000021
1.1	0.637495	0.125225	0.173490	0.631601	0.484792	0.591273	0.430660	
1.2	0.464117	0.525968	0.155123	0.348160	0.495543	0.522456	0.550613	
1.3	0.493748	0.639962	0.371967	0.599096	0.470965	0.628861	0.600373	
1.4	0.507620	0.683722	0.283558	0.383126	0.489919	0.558897	0.517625	
1.5	0.591945	0.545211	0.509431	0.581511	0.531687	0.571069	0.622994	
1.6	0.539823	0.440606	0.481209	0.615607	0.503650	0.366996	0.534068	
1.7	0.541962	0.397654	0.495012	0.686918	0.456253	0.497463	0.671947	
1.8	0.558370	0.562000	0.629863	0.232478	0.405986	0.554857	0.522868	
1.9	0.474184	0.393826	0.766656	0.408858	0.507267	0.629427	0.626592	

	3.1		3.1		3.1
2.1	−0.000005	2.6	0.000028	2.11	−0.000004
2.2	−0.000026	2.7	0.000033	2.12	−0.000035
2.3	0.000043	2.8	−0.000064	2.13	−0.000007
2.4	0.000016	2.9	−0.000007	2.14	−0.000026
2.5	0.000016	2.10	−0.000021	2.15	0.000023

Table 8. Weights and biases of the (5:1:1:1)MLP in (12:1:1)MLP&(5:1:1:1)MLP hybrid model.

	2.1	3.1	4.1
Tresh	0.318541	−0.055030	−0.63998
1.1	−0.089149		
1.2	0.180931		
1.3	0.263460		
1.4	0.239402		
1.5	0.060785		
2.1		−0.435689	
3.1			−1.29577

The (12:1:1)MLP&(6:9:1)RBFN showed better performance among the MLP&RBFN hybrid models. The parameters of the appropriated (6:9:1) RBFN are given in table 9.

The (12:48:1)RBFN&(12:8:8:1)MLP and the (12:48:1)RBFN&(12:12:1)RBFN hybrid models showed better performance among the RBFN&RBFN and the RBFN&MLP hybrid models, respectively. The parameters of the (12:8:8:1)MLP and the (12:12:1)RBFN networks are given, respectively, in tables 10 and 11.

3.5 *Comparisons*

The observed and forecasted values of the number of monthly tourist arrivals to Turkey between the January 2002 and December 2003 period are given in table 12 in order to compare the

Table 9. Weights and biases of the (6:9:1)RBFN in (12:1:1)MLP&(6:9:1)RBFN hybrid model.

	2.1	2.2	2.3	2.4	2.5
Thresh	3.636305	3.598036	4.075343	2.455933	2.626410
1.1	0.719707	0.633418	0.480344	0.646147	0.438251
1.2	0.625472	0.697336	0.606274	0.346976	0.527577
1.3	0.642923	0.614665	0.571926	0.571909	0.658577
1.4	0.561171	0.614321	0.672393	0.577481	0.347993
1.5	0.617908	0.590268	0.453837	0.339249	0.486000
1.6	0.524186	0.728897	0.647109	0.430076	0.532385
	2.6	2.7	2.8	2.9	3.1
Thresh	3.532489	1.788779	1.119374	2.339018	0.000032
1.1	0.589295	0.579048	0.721618	0.419984	
1.2	0.435030	0.587858	0.801243	0.613162	
1.3	0.592224	0.181142	0.907129	0.395466	
1.4	0.653027	0.528050	0.859107	0.722138	
1.5	0.732899	0.506009	0.885713	0.739940	
1.6	0.557442	0.379182	0.848770	0.751394	
	3.1				
2.1	0.000005				
2.2	−0.000030				
2.3	0.000000				
2.4	0.000023				
2.5	0.000006				
2.6	−0.000014				
2.7	−0.000002				
2.8	0.000059				
2.9	0.000009				

performances of the ARIMA, ANN, and hybrid models mentioned in previous sections. The graph of forecasted values is given in figure 2.

The MSE (RMSE), MAE, and MAPE values for the defined models are given in table 13. These measures are directly used to compare known models.

As seen in table 13, owing to the MSE (RMSE) measure, (12:1:1)MLP&(6:9:1)RBFN and (12:1:1)MLP&(5:1:1:1)MLP hybrid models and (12:1:1)MLP ANN models, and owing to the MAE measure (12:1:1)MLP&(5:1:1:1)MLP, (12:1:1)MLP&(6:9:1)RBFN hybrid and (12:1:1)MLP ANN models, make the best forecasts. It can be seen that because of the MSE (RMSE) and MAE measures, the hybrid models (12:1:1)MLP&(5:1:1:1)MLP and (12:1:1)MLP&(6:9:1)RBFN and the (12:1:1)MLP ANN model make better forecasts when compared with other selected models.

It is concluded that because of the MAPE measure ARIMA(1,1,1)(1,1,0)$_{12}$&(6:8:1)MLP, the (12:48:1)RBFN&(12:8:8:1)MLP and (12:1:1)MLP&(6:9:1)RBFN hybrid models respectively make better forecasts.

According to these results, similar models are defined as models with the best performances because of the MSE and MAE measures. However, because of the MAPE measure, the ARIMA(1,1,1)(1,1,0)$_{12}$&(6:8:1)MLP and (12:48:1)RBFN&(12:8:8:1)MLP hybrid models are the models with better performance than the other selected models.

Among the models with one component (the models that are not hybrid models), because of the MSE and MAE measures, the (12:1:1)MLP showed the best performance and the (12:48:1)RBFN showed the worst performance. It is known that the RBFN is usually unsuccessful in extrapolation problems [31]. Thus, the result from table 13 for the RBFN is an expected one. Although the RBFN makes bad forecasts by itself, as can be seen from table 13, when it is used as a second component, the (12:1:1)MLP&(6:9:1)RBFN hybrid model showed

Table 10. Weights and the biases of the (12:8:8:1)MLP in the (12:48:1)RBFN&(12:8:8:1)MLP hybrid model.

	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	
Thresh	−0.615281	−0.946041	0.088989	0.412924	0.674040	0.360338	0.081607	−0.189698	
1.1	0.649558	0.139388	0.734809	0.567895	−0.977693	0.064431	−0.252522	0.582959	
1.2	0.062171	−0.871867	0.228383	−0.256319	−0.284524	−0.723271	0.458857	−0.371907	
1.3	−0.366780	0.230264	0.589741	0.233112	−0.543476	−0.303465	−0.792526	−0.928245	
1.4	0.267498	−0.156803	−0.648617	0.507572	0.818787	−0.032901	0.722582	0.409957	
1.5	−0.000533	−0.543670	−0.666306	−0.380854	0.233241	0.472763	−0.672493	0.369827	
1.6	−0.546201	0.245034	−0.590576	0.896252	0.128665	−0.431440	−0.507286	−0.578261	
1.7	0.913671	0.806266	0.921900	0.957143	−0.399189	0.041015	−0.319767	−0.767415	
1.8	−0.162613	−0.508579	−0.058484	0.820520	0.926778	0.504357	−0.325828	0.635759	
1.9	0.123262	−0.365327	−0.055822	−0.837163	−0.377446	0.736090	0.677681	−0.023240	
1.10	0.051004	−0.383909	0.966997	−0.728461	−0.633879	−0.656685	0.092391	−0.791676	
1.11	−0.497448	0.113802	0.017110	0.780756	−0.418521	0.060514	−0.733188	0.382616	
1.12	0.234422	0.608157	−0.170772	−0.219141	−0.322876	−0.170464	0.551308	−0.270666	
	3.1	3.2	3.3	3.4	3.5	3.6	3.7	3.8	4.1
Thresh	−0.010395	−0.057840	0.576224	−0.082142	0.71019	0.276618	0.656145	−0.120284	−0.894716
2.1	0.591370	0.066678	−0.112148	0.494170	−0.15215	−0.020541	0.159603	−0.897964	
2.2	0.185381	0.990531	0.790876	−0.778926	−1.04705	−0.846423	−0.389086	0.791972	
2.3	0.893992	−0.057135	0.588636	−0.212809	−0.56051	−0.324198	0.612532	−0.881267	
2.4	0.005973	0.362380	−0.686853	0.856398	0.26046	−0.124689	0.797130	0.178243	
2.5	0.814362	−0.077315	0.467994	0.500403	0.39091	0.853966	0.719465	−0.078072	
2.6	0.104380	0.732808	0.509218	−0.077488	−0.51767	−0.872448	−0.219228	−0.822140	
2.7	0.638234	−0.252373	−0.656206	−0.201921	−0.90630	0.081861	0.146502	−0.906984	
2.8	0.900164	−0.494313	−0.683631	0.221736	−0.42602	−0.525631	0.763561	−0.999927	
3.1									0.463455
3.2									0.021859
3.3									−0.238889
3.4									−0.615665
3.5									0.853993
3.6									−0.467848
3.7									0.550647
3.8									0.218893

Table 11. Weights and biases of the (12:12:1)RBFN in the (12:48:1)RBFN&(12:12:1)RBFN hybrid model.

	2.1	2.2	2.3	2.4	2.5	2.6	
Thresh	7.751850	2.589252	8.842412	2.300625	8.842412	4.725020	
1.1	0.326919	0.436297	0.363383	0.333748	0.333175	0.413955	
1.2	0.544137	0.301394	0.401072	0.335402	0.354321	0.351177	
1.3	0.390504	0.330452	0.339969	0.528836	0.334450	0.426005	
1.4	0.303067	0.459942	0.437463	0.217360	0.325987	0.409826	
1.5	0.416708	0.364712	0.452168	0.318248	0.330103	0.350253	
1.6	0.420093	0.322948	0.318741	0.418243	0.301328	0.561672	
1.7	0.300643	0.664499	0.416244	0.092590	0.334769	0.522613	
1.8	0.439038	0.328469	0.433674	0.364579	0.317930	0.250786	
1.9	0.403719	0.644205	0.294395	0.046012	0.332181	0.352519	
1.10	0.349112	0.426632	0.406400	0.418615	0.345290	0.380620	
1.11	0.497448	0.405701	0.351095	0.357152	0.333927	0.321967	
1.12	0.358103	0.549054	0.306449	0.393950	0.323635	0.579267	
	2.7	2.8	2.9	2.10	2.11	2.12	3.1
Thresh	2.289151	5.449336	7.811204	6.859481	8.824874	1.582496	−0.000032
1.1	0.491956	0.575126	0.418363	0.378747	0.365916	0.378446	
1.2	0.367271	0.407461	0.298132	0.350592	0.356343	0.366921	
1.3	0.338171	0.347620	0.384516	0.527150	0.418409	0.867701	
1.4	0.685064	0.388390	0.321167	0.437263	0.362048	0.297833	
1.5	0.306814	0.398572	0.326701	0.248014	0.383109	0.628251	
1.6	0.691927	0.344431	0.379218	0.365948	0.463666	0.397127	
1.7	0.394290	0.436004	0.364569	0.356491	0.364621	0.444465	
1.8	0.390801	0.433152	0.390697	0.339418	0.382158	0.363447	
1.9	0.555662	0.356582	0.403191	0.460620	0.411180	0.483547	
1.10	0.453078	0.576932	0.167584	0.344997	0.375163	0.352332	
1.11	0.431594	0.369323	0.378211	0.361377	0.388093	0.507714	
1.12	0.440899	0.391348	0.340669	0.524742	0.378278	0.591160	
	3.1		3.1		3.1		3.1
2.1	−0.000015	2.4	−0.000037	2.7	−0.000014	2.10	0.000013
2.2	−0.000024	2.5	−0.000002	2.8	0.000043	2.11	0.000021
2.3	−0.000009	2.6	−0.000011	2.9	0.000003	2.12	−0.000034

the best performance when it is used as a second component. When only the RBFN is used, the number of radial units is defined as 48. In other words, when the start data set is used in education and forecast, 48 radial centers are selected. In the (12:1:1)MLP&(6:9:1)RBFN hybrid model, when the second component RBFN is applied to the remaining data, the number of radial centers are decreased to nine. This situation can be explained by the fact that the differences between the residuals are getting smaller.

Owing to the MAPE measure, among the models with one component, the ARIMA(1,1,1)(1,1,0)$_{12}$ showed better performance than the other models with one component.

Depending on the comparisons in this section, we can conclude that there is no certain measure in selecting the best model. The discussions in literature on this subject support this idea [42, 43]. Hence, the performance of any model in this study is evaluated by a certain performance measure.

4. Conclusions

It is seen that hybrid models also have good performance in forecasting, besides the ARIMA and the ANN in analysing time series. The hybrid models are composites of the same kinds of model or different kinds of model. In general, it is recommended that one component of a

Table 12. Observed and forecasted values for the January 2002–December 2003 period.

	Forecasted number of tourists					
Observed number of tourists	ARIMA (1,1,1)(1,1,0)12	(12:1) Linear NN	(12:1:1)MLP	(12:48:1)RBFN	ARIMA (1,1,1)(1,1,0) &(6:8:1)MLP	ARIMA (1,1,1)(1,1,0) &(6:7:1)RBFN
306,597	367,821	158,076	191,260	400,331	347,993	334,042
426,405	412,774	348,196	355,334	473,596	461,410	407,279
675,687	547,564	569,893	555,900	538,181	550,675	541,688
852,930	899,773	880,450	867,338	829,218	876,473	888,106
1,325,752	1,253,372	1,336,921	1,348,072	1,148,843	1,269,612	1,264,744
1,457,615	1,405,884	1,527,239	1,588,867	1,363,436	1,398,638	1,413,286
1,897,112	1,864,143	1,831,861	1,883,276	1,736,942	1,885,077	1,875,040
1,900,120	1,701,509	1,752,364	1,800,169	1,624,099	1,734,927	1,721,460
1,770,566	1,567,816	1,569,988	1,603,643	1,405,547	1,583,086	1,581,068
1,420,386	1,219,301	1,213,457	1,248,384	1,237,010	1,244,571	1,238,147
662,985	605,193	685,096	721,147	684,957	626,594	624,821
559,873	450,663	423,025	456,867	378,877	465,627	467,431
363,983	405,529	140,579	217,579	395,642	435,115	426,456
481,252	455,805	258,948	311,306	490,830	480,857	474,399
499,663	608,619	503,928	532,325	541,329	635,547	627,504
669,288	995,138	876,049	881,239	737,517	1,026,223	1,015,495
1,146,309	1,386,130	1,385,355	1,409,756	1,107,727	1,411,410	1,404,582
1,510,951	1,557,352	1,664,131	1,744,052	1,312,818	1,585,067	1,577,125
2,130,949	2,042,219	1,963,100	2,027,821	1,696,669	2,071,592	2,061,750
2,275,055	1,856,667	1,918,087	1,995,540	1,720,924	1,883,812	1,875,594
1,874,329	1,697,800	1,717,861	1,789,345	1,438,117	1,728,513	1,717,649
1,657,726	1,299,698	1,352,890	1,434,308	1,301,649	1,329,667	1,318,860
776,181	641,975	815,274	896,678	848,043	671,059	661,332
643,872	481,979	466,458	532,850	406,853	512,699	501,585

(*continued*)

Table 12. Continued.

Observed number of tourists	Forecasted number of tourists					
	(12:1)LinearNN& (6:8:1)MLP	(12:1)LinearNN& (9:15:1)RBFN	(12:1:1)MLP& (5:1:1:1)MLP	(12:1:1)MLP& RBFN(6:9:1)	(12:48:1)RBFN& (12:8:8:1)MLP	(12:48:1)RBFN& (12:12:1)RBFN
306,597	170,546	185,745	230,910	249,187	341,419	410,067
426,405	360,952	278,269	390,367	438,835	499,455	449,964
675,687	584,345	532,764	574,744	611,032	557,715	572,640
852,930	897,119	848,922	883,799	918,581	865,147	840,541
1,325,752	1,353,273	1,355,597	1,365,567	1,399,967	1,201,416	1,150,715
1,457,615	1,545,780	1,535,491	1,602,650	1,637,208	1,473,891	1,392,297
1,897,112	1,853,048	1,861,066	1,895,311	1,923,804	1,735,975	1,725,590
1,900,120	1,773,985	1,761,293	1,812,367	1,839,383	1,666,534	1,651,394
1,770,566	1,592,956	1,615,192	1,615,288	1,636,300	1,440,349	1,423,161
1,420,386	1,237,230	1,229,698	1,259,347	1,276,694	1,189,536	1,222,406
662,985	709,202	737,050	732,045	744,832	667,341	664,780
559,873	448,202	464,216	467,693	477,234	329,979	395,196
363,983	166,444	198,709	228,229	233,096	356,760	405,644
481,252	285,226	298,961	321,889	324,065	465,878	472,984
499,663	530,830	565,158	542,889	541,992	507,598	563,654
669,288	903,274	926,664	891,774	888,678	726,540	742,049
1,146,309	1,412,894	1,444,658	1,420,264	1,414,926	1,111,982	1,106,195
1,510,951	1,692,097	1,716,038	1,754,555	1,747,994	1,354,913	1,338,086
2,130,949	1,991,319	2,025,626	2,038,317	2,030,353	1,707,417	1,698,739
2,275,055	1,946,558	1,974,114	2,006,027	1,997,233	1,768,917	1,731,824
1,874,329	1,746,588	1,777,735	1,799,829	1,790,256	1,471,399	1,464,096
1,657,726	1,381,775	1,410,833	1,444,791	1,434,807	1,316,208	1,308,556
776,181	844,338	877,208	907,158	896,683	861,066	840,884
643,872	495,689	525,032	543,329	532,631	392,354	433,343

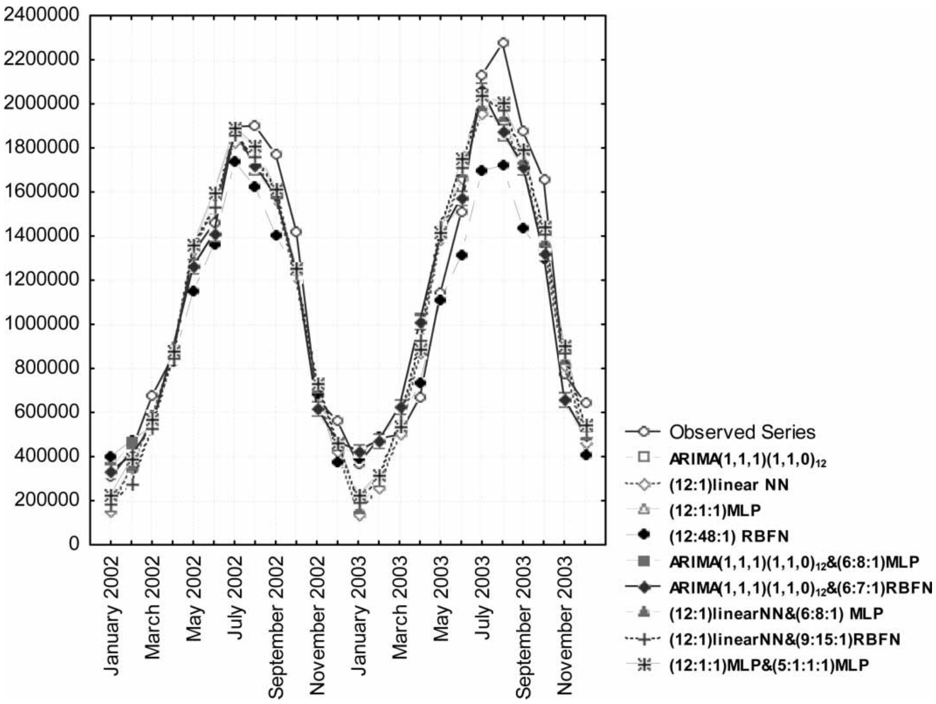

Figure 2. Observed and forecasted values for the January 2002–December 2003 period.

Table 13. Performance values for the selected models.

Models	MSE	RMSE	MAE	MAPE
ARIMA(1,1,1)(1,1,0)$_{12}$	3.06E+10	174,926.3	137,589.4	13.74
(12:1)LinearNN	2.93E+10	171,046.8	144,662.4	17.06
(12:1:1)MLP	2.19E+10	147,909.3	127,838.8	14.96
(12:48:1)RBFN	5.40E+10	232,280.8	176,592.2	14.92
ARIMA(1,1,1)(1,1,0)$_{12}$&(6:8:1)MLP	2.79E+10	167,068.6	128,148.6	13.15
ARIMA(1,1,1)(1,1,0)$_{12}$&(6:7:1)RBFN	2.86E+10	169,058.0	129,721.5	12.95
(12:1)LinearNN&(6:8:1)MLP	2.73E+10	165,220.0	141,377.4	16.59
(12:1)LinearNN&(9:15:1)RBFN	2.57E+10	160,479.0	139,907.8	16.97
(12:1:1)MLP&(5:1:1:1)MLP	2.09E+10	144,410.7	123,104.7	13.92
(12:1:1)MLP&(6:9:1)RBFN	2.07E+10	143,958.1	123,184.7	13.45
(12:48:1)RBFN&(12:8:8:1)MLP	4.78E+10	218,540.3	160,724.3	13.26
(12:48:1)RBFN&(12:12:1)RBFN	4.99E+10	223,470.6	167,694.1	14.12

hybrid model be linear and the other nonlinear [11]. The reason for this is that, in the nature of real-life data, both linear and nonlinear relationships can occur.

In this study, the (12:1:1)MLP&(5:1:1:1)MLP and (12:1:1)MLP&(6:9:1)RBFN hybrid models with two nonlinear components have shown better performance when compared with other models in forecasting the number of monthly tourist arrivals to Turkey owing to the MSE (RMSE) and the MAE measures. This could change the idea that one component of the hybrid model should be linear. Depending on the findings in this study, it can be concluded that the hybrid ANN models with two nonlinear components can show better performance for appropriate forecasting problems.

Although the RBFN showed poor performance in extrapolation problems, it can help to obtain a hybrid model with a good performance when used as a component in hybrid models.

The best model should be chosen depending on the statistical and the mathematical structure of the data set. Unfortunately, there is no certain measure or test for the selection of an appropriate model because of the nature of the data. The numerous experimental studies that will be done on this subject can provide a definite and useful aggregation on a criterion.

References

[1] Zhang, G.P., Patuwo, E.B. and Hu, M.Y., 2001, A simulation study of artificial neural networks for nonlinear time-series forecasting. *Computers Operations Research*, **28**, 381–396.

[2] Box, G.P. and Jenkins, G.M., 1970, *Time Series Analysis, Forecasting and Control* (San Francisco, CA: Holden-Day).

[3] Granger, C.W.J. and Terasvirta, T., 1993, *Modelling Nonlinear Economic Relationships* (Oxford: Oxford University Press).

[4] Ansuj, A.P., Camargo, M.F., Radharamanan, R. and Petry, D.G., 1996, Sales forecasting using time series and neural networks. *Computational and Industrial Engineering*, **31**(1), 421–424.

[5] Caire, D., Hatabian, G. and Muller, C., 1992, Progress in forecasting by neural networks. *International Conference on Neural Networks 2*, Baltimore, MD, USA, 540–545.

[6] Chin, K. and Arthur, R., 1996, Neural network vs. conventional methods of forecasting. *Journal of Business Forecasting*, **14**(4), 17–22.

[7] Hill, T., O'Connor, M., Remus, W., 1996, Neural network models for time series forecasts. *Management Science*, **42**(7), 1082–1092.

[8] Kohzadi, N., Boyd, M.S., Kermanshahi, B. and Kaastra, I., 1996, A comparison of artificial neural network and time series model for forecasting commodity prices. *Neurocomputing*, **10**(2), 169–181.

[9] Maier, H.R. and Dandy, G.C., 1996, Neural network models for forecasting univariate time series. *Neural Networks World*, **6**(5), 747–772.

[10] Tseng, F.M., Yu, H.C. and Tzeng, G.H., 2002, Combining neural network model with seasonal time series ARIMA model. *Technological Forecasting & Social Change*, **69**, 71–87.

[11] Zhang, G.P., 2003, Time-series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, **50**, 159–175.

[12] Bates, J.M. and Granger, C.W.J., 1969, The combination of forecasts. *Operational Research Quarterly*, **20**, 451–468.

[13] Reid, D.J., 1968, Combining three estimates of gross domestic product. *Economica*, **35**, 431–444.

[14] Makridakis, S., Anderson, A., Carbone, R., Fildes, R., Hibdon, M., Lewandowski, R., Newton, J. and Winkler, R., 1982, The accuracy of extrapolation (time series) methods: results of a forecasting competition. *Journal of Forecasting*, **1**, 111–153.

[15] Clemen, R., 1989, Combining forecasts: a review and annotated bibliography with discussion. *International Journal of Forecasting*, **5**, 559–608.

[16] Makridakis, S., 1989, Why combining works? *International Journal of Forecasting*, **5**, 601–603.

[17] Newbold, P. and Granger, C.W.J., 1974, Experience with forecasting univariate for time series and the combination of forecasts (with discussion). *Journal of Royal Statistical Society Series A*, **137**, 131–164.

[18] Palm, F.C. and Zellner, A., 1992, To combine or not to combine? Issues of combining forecasts. *Journal of Forecasting*, **11**, 687–701.

[19] Winkler, R., 1989, Combining forecasts: a philosophical basis and some current issues. *International Journal of Forecasting*, **5**, 605–609.

[20] Ginzburg, I. and Horn, D., 1994, Combined neural networks for time series analysis. *Advances in Neural Information Processing Systems*, **6**, 224–231.

[21] Pelikan, E., Groot, C.D. and Wurtz, D., 1992, Power consumption in west Bohemia: improved forecasts with decorrelating connectionist networks. *Neural Networks World*, **2**, 701–712

[22] Wedding, D.K. and Cios, K.J., 1996, Time-series forecasting by combining RBF networks, certainty factors, and the Box–Jenkins model. *Neurocomputing*, **10**, 149–168.

[23] Voort, V.D., Dougherty, M. and Watson, M., 1996, Combining Kohonen maps with ARIMA time series models to forecast traffic flow. *Transport Research Circulation* (Energe Technol.), **4C**(5), 307–318.

[24] Luxhoj, J.T., Riis, J.O. and Stensballe, B., 1996, A hybrid econometric-neural-network modelling approach for sales forecasting. *International Journal of Production Economics*, **43**, 175–192.

[25] Wang, J.H. and Leu, J.Y., 1996, Stock market trend prediction using ARIMA-based neural networks. *IEEE International Conference on Neural Networks*, Washington DC, USA, **4**, 2160–2165.

[26] Su, C.T., Tong, L.I. and Leou, C.M., 1997, Combination of time series and neural network for reliability forecasting modelling. *Journal of the Chinese Institute of Industrial Engineering*, **14**(4), 419–429.

[27] Law, R. and Au, N., 1999, A neural networks model to forecast Japanese demand for travel to Hong Kong. *Tourism Management*, **20**(1), 89–97.

[28] Law, R., 2000, Back-propagation learning in improving the accuracy of neural network-based tourism demand forecasting. *Tourism Management*, **21**, 331–340.

[29] Cho, V., 2003, A comparison of different approaches to tourism arrival forecasting. *Tourism Management*, **24**, 323–330.

[30] Kim, J., Wei, S. and Ruys, H., 2003, Segmenting the market of West Australian senior tourists using an artificial neural network. *Tourism Management*, **24**, 25–34.
[31] Bishop, C.M., 1995, *Neural Networks for Pattern Recognition* (Oxford: Clarenden Press).
[32] Haykin, S., 1999, *Neural Networks: A Comprehensive Foundation* (Prentical Hall).
[33] Zhang, G.P., Patuwo, E.B. and Hu, M.Y., 1998, Forecasting with artificial neural networks: the state of the art. *International Journal of Forecasting*, **14**, 35–62.
[34] Rumelhart, D.E., Hinton, G.E. and Willams, R.J., 1986, Learning representation by backpropagating errors. *Nature*, **323**(6188), 533–536.
[35] Bhaya, A. and Kaszkurewicz, E., 2004, Steepest descent with momentum for quadratic functions is a version of the conjugate gradient method. *Neural Networks*, **17**, 65–71.
[36] Qian, N., 1999, Efficient backpropagation learning using optimal learning rate and on the momentum term in gradient descent learning algorithms. *Neural Networks*, **102**(1), 145–151.
[37] Yu, X.H. and Chen, G.A., 1997, Efficient backpropagation learning using optimal learning rate and momentum. *Neural Networks*, **10**(3), 517–527.
[38] Nocedal, J. and Wright, S.J., 1999, *Numerical Optimization* (New York: Springer-Verlag).
[39] Moller, M., 1993, A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, **6**(4), 525–533.
[40] Granger, C.W.J., 1989, Combined forecasts – Twenty years later. *Journal of Forecasting*, **8**, 167–173.
[41] Perrone, M.P. and Cooper, L., 1993, When networks disagree: ensemble method for hybrid neural networks. In: R.J. Mammone (Ed.) *Neural Networks for Speech and Image Processing* (London: Chapman & Hall), pp. 126–142.
[42] Zhang, G.P. and Qi, M., 2005, Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research*, **160**, 501–514.
[43] Zhang, G.P., 2001, An investigation of neural networks for linear time-series forecasting. *Computers and Operations Research*, **28**, 1183–1202.
[44] Clements, M.P. and Nendry, D.F., 1993, On the limitations of comparing mean square forecast errors. *Journal of Forecasting*, **12**, 617–637.
[45] www.die.gov.tr