Sains Malaysiana 51(8)(2022): 2645-2654
http://doi.org/10.17576/jsm-2022-5108-23

Performance of Levenberg-Marquardt Neural Network Algorithm in Air Quality Forecasting

(Prestasi Algoritma Rangkaian Neuron Levenberg-Marquardt dalam Ramalan Kualiti Udara)

CHO KAR MUN[1], NUR HAIZUM ABD RAHMAN[1,*] & ISZUANIE SYAFIDZA CHE ILIAS[2]

[1]Department of Mathematics and Statistics, Faculty of Science, Universiti Putra Malaysia 43400 UPM Serdang, Selangor Darul Ehsan, Malaysia
[2]Institute for Mathematical Research, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor Darul Ehsan, Malaysia

Received: 2 July 2021/Accepted: 30 January 2022

ABSTRACT

Levenberg-Marquardt algorithm and conjugate gradient method are frequently used for optimization in multi-layer perceptron (MLP). However, both algorithms have mixed conclusions in optimizing MLP in time series forecasting. This study uses autoregressive integrated moving average (ARIMA) and MLP with both Levenberg-Marquardt algorithm and conjugate gradient method. These methods were used to predict the Air Pollutant Index (API) in Malaysia's central region where represent urban and residential areas. The performances were discussed and compared using the mean square error (MSE) and mean absolute percentage error (MAPE). The result shows that MLP models have outperformed ARIMA models where MLP with Levenberg-Marquardt algorithm outperformed the conjugate gradient method.

Keywords: Algorithm; ARIMA; artificial neural network; forecasting; multi-layer perceptron

ABSTRAK

Algoritma Levenberg-Marquardt dan kaedah kecerunan konjugat sering digunakan untuk pengoptimuman dalam *perceptron* pelbagai lapisan (MLP). Walau bagaimanapun, kedua-dua algoritma mempunyai kesimpulan yang berbeza dalam mengoptimumkan ramalan siri masa menggunakan MLP. Kajian ini menggunakan purata bergerak bersepadu autoregresif (ARIMA) dan MLP dengan kedua-dua algoritma Levenberg-Marquardt dan kaedah kecerunan konjugat. Kaedah ini digunakan untuk meramalkan Indeks Pencemaran Udara (IPU) di wilayah tengah Malaysia yang mewakili kawasan bandar dan kediaman. Prestasi dibincang dan dibandingkan dengan menggunakan ralat kuasa dua min (MSE) dan ralat peratusan mutlak (MAPE). Hasilnya menunjukkan bahawa model MLP telah mengatasi model ARIMA dengan MLP dan algoritma Levenberg-Marquardt mengatasi kaedah kecerunan konjugat.

Kata kunci: Algoritma; ARIMA; *perceptron* pelbagai lapisan; ramalan; rangkaian neuron tiruan

INTRODUCTION

Artificial Neural Networks (ANN) can be defined as artificial adaptive systems modelled after the human brain's cerebral cortex's functioning processes. ANN learns the behaviour from past data, generates a learnt pattern (synaptic weight), and solves the problem using the taught pattern (Saratha et al. 2020). ANN is formed from numerous processing elements known as nodes or neurons connected to coefficients or weights (De Cosmi et al. 2020). In each processing element, the arriving signals' inputs are multiplied by the connection weights and summed. It is then passed through a transfer or activation function which is the weighed sum of the neuron's inputs. The transfer function that is often used is the sigmoid function because the sigmoid function enables a smooth transition between input and output (Chen et al. 2006). Finally, an output is produced.

In ANN, the network architectures are determined by the type of connections between the neurons, and there are two main categories: feed forward neural network and recurrent neural network (Zhu et al. 2020). A feed forward neural network does not receive feedback from the outputs of the neurons to the input neurons. It is a static system as only a set of output values are generated, and it does not have memory since the response of the input is independent of the initial network condition (Jain et al. 1996). In contrast, a recurrent neural network, a dynamic system, receives feedback or a connection from the output to input neurons. The inputs are further adjusted as there are feedbacks, and the network is directed to a new state. Many of the exceedingly tough difficulties presented by water sciences and hydrology have been addressed with ANN (Mahmoud et al. 2016). Besides, ANNs are widely utilized in a variety of areas such as business, industry, science and further used as a forecasting tool. Forecasting is reliable and effective in control measures, and thus, it can be suggested as a preventive and evasive action on what regulations are to be enforced (Nurulilyana et al. 2011). It is due to ANNs properties which are data-driven self-adaptive methods, as they can learn from examples and recognize some functional relationships among the data (Zhang et al. 1998). Besides, generalization can be made through ANNs after the data have been trained and correctly generalized to brand-new data.

The ANN must undergo a learning process in which the network architecture and connection weights are constantly updated for the network to operate a particular task efficiently. Among the learning paradigms are supervised and unsupervised learning. Supervised learning involves both inputs and correct outputs, while unsupervised learning does not involve output with every input provided. An example of a supervised learning model is the feed-forward or Multi-Layer Perceptron model, whereas unsupervised learning is Self-organizing neural networks (Sathya & Abraham 2013).

In supervised learning, it is necessary to estimate the loss and compare the prediction result concerning the correct result provided and measured. Therefore, the backpropagation learning rule is frequently used to train the network as the loss is propagated backward throughout the network's layers and the weights are continuously adjusted until good prediction results are attained (Sheela et al. 2013). The backpropagation training algorithms aim to minimize the loss to zero as close as possible. Some instances of the training algorithms used are Gradient Descent, Conjugate Gradient, Quasi-Newton, Resilient Backpropagation, and Levenberg-Marquardt algorithms (Zafer & Adnan Fatih 2017).

Multi-Layer Perceptron (MLP) is one of the most potential models that has been proposed in time series forecasting. MLP is comprised of several layers of nodes; input layer, hidden layer, and output layer. Past observations of the time series will enter the input layer, and future values will be yielded by the output layer (An & Anh 2015). A supervised learning MLP model will be developed in this study as its structure is well known for forecasting purposes. The MLP will be trained with the Levenberg-Marquardt algorithm and conjugate gradient descent. Within the MLP, the number of hidden layers to be used as the neural network with one hidden layer can yield an arbitrarily close approximation to any continuous nonlinear mapping as presented by literature in Kavzoglu (1999).

Materials and Methods

The Monitoring Station

The air quality data from the Putrajaya monitoring station in 2015 was used. The data was divided into two data sets; a training data set from January to December 2015 to identify the API model and a testing data set in January 2016 to check the model performance. The air quality data set from this study was obtained from the Department of Environment (DoE), Malaysia. Putrajaya is one of the three federal territories in Malaysia. It is a planned city and the federal administrative center of the Malaysian capital starting in 1999 due to overcrowding and congestion in city center Kuala Lumpur and classified as an urban area located in the central region of Malaysia.

ARIMA

Box-Jenkins's methodology is a three-step iterative approach in building an autoregressive integrated moving average (ARIMA) model, which includes model identification, parameter estimation, and diagnostic checking. For a given time series, it is crucial to develop a best-fitted model and, at the same time to consider the principle of parsimony. The parsimonious model is often the model with the smallest possible number of parameters that can preserve an adequate and accurate representation of the time-series data. The Bayesian Information Criterion (BIC) is used as a measure of model identification. BIC proposed a penalty term for the number of parameters to overcome overfitting due to increased likelihood when the parameter rises (Evans 2019). The definition of BIC is given as follows:

$$BIC(p) = n \ln n \left(\frac{\hat{\sigma}_e^2}{n}\right) + p + p \ln n \quad (1)$$

where n is the number of observations; p is the number of parameters in the model and $\hat{\sigma}_e^2$ is the sum of squared residuals. The optimal model order is selected based on the number of parameters that give the minimum BIC. The lower the value of BIC, the more likely the model is the true model.

ARIMA model is a generalization of autoregressive moving average (ARMA) models of non-stationary time series. The application of ARMA models to non-stationary time series in practice is often unsatisfactory. ARIMA models make use of the differencing procedure to make the time series stationary. The differencing value is the difference between the current and the previous time. ARIMA (p,d,q) model consists of several components, which include AR (p), I (d), and MA (q). The ARIMA model can be written in backshift notation as:

$$\phi(B)(1-B)^d Y_t = c + \theta(B)\varepsilon_t \quad (2)$$

where B is the backshift operator; $\phi(B) = 1 - \phi_1 B^1 - \phi_2 B^2 - \cdots - \phi_p B^p$, and $\theta(B) = 1 - \theta_1 B^1 - \theta_2 B^2 - \cdots - \theta_q B^q$.

NEURAL NETWORK

In artificial neural networks, data set are divided into three sets named training, testing, and validation sets. The training set in which the neural network will learn the patterns displayed in the data. The testing set has a range of 10% to 30% of the training set (Sulafa Hag 2014). The purpose of the testing set is to assess a trained network on its ability to generalize. From here, the neural network(s) with the best performance on the testing set will be chosen. Finally, the validation set is used to verify the trained network's performance.

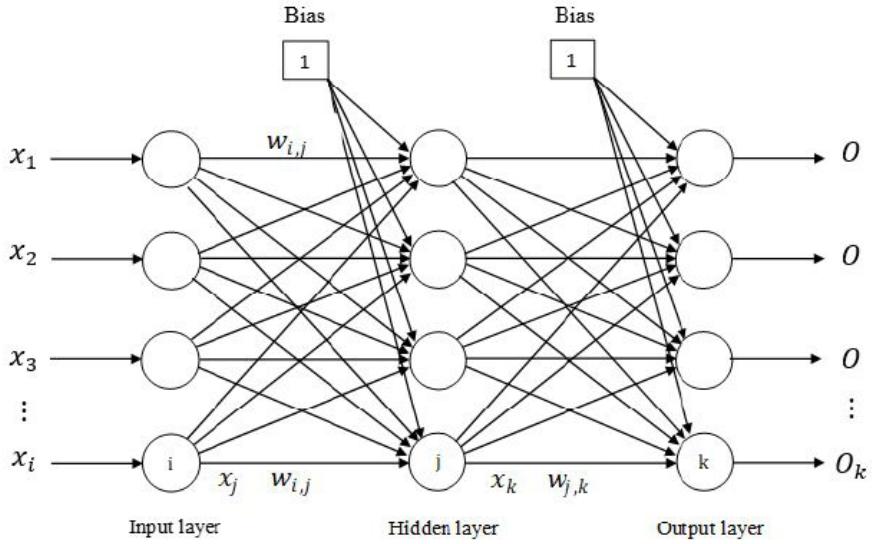


FIGURE 1. Structure of a multi-layer perceptron

Figure 1 shows the structure of a MLP, which consists of three layers. The first layer of the neural network is the input layer, where the past observations of the time series are passed through. It communicates with the external environment, and a pattern is introduced to the neural network. The hidden layer is the intermediate layer that separates the input layer and the output layer. One hidden layer in the neural network with an adequate number of hidden neurons is competent for ANNs to generalize any continuous function (Dong et al. 2013). In increasing the number of hidden layers, the neural network will risk overfitting where each point is memorized instead of learning the general patterns. The time required for computations will also be longer. The output layer of the neural network will yield a future value to a time series problem.

A transfer function is a mathematical formula such the output of the processing neuron is calculated (Kaastra & Boyd 1996). It is put into use to avoid outputs from getting larger as the neural network can paralyze.

Therefore, the training is obstructed. The sigmoid function is the regular transfer function used for data involving time series, considering it is nonlinear and continuously differentiable for network learning. The sigmoid function is as follows:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

The MLP model with a set of P input variables can be expressed through the formula:

$$Y = f\left\{w_{co} + \sum_{h} w_{ho} f\left(w_{ch} + \sum_{i} w_{ih} x_{t-j}\right)\right\} \tag{4}$$

where w_{ch} is the weights for constant input and hidden neurons; w_{co} is the weights of constant input and output; w_{ih} is the weights between inputs and hidden neurons; w_{ho} is the weights between hidden neurons and output; and f is the activation function. The stopping criteria can be divided into two, late stopping and early stopping. The idea of late stopping is that the neural network is trained as far as a particular error is attained (Dong et al. 2013). However, late stopping can lead to overfitting. Early stopping is effective in preventing overfitting, and it has been reported to be easily understood and is excellent in regularization (Lodwich et al. 2009).

LEVENBERG-MARQUARDT ALGORITHM

The Levenberg-Marquardt algorithm combines the gradient descent algorithm and Gauss-Newton method and serves as an intermediate optimization algorithm in neural network training (Kermani et al. 2005). The gradient descent algorithm applies the first-order derivative of the total error function to obtain the minimum in the error space. The gradient, g can be defined as:

$$g = \frac{\partial E(x,w)}{\partial w} = \left[\frac{\partial E}{\partial w_1} \; \frac{\partial E}{\partial w_2} \; \cdots \; \frac{\partial E}{\partial w_N}\right]^T \tag{5}$$

where $E(x,w)$ is the sum of square error function and is computed by:

$$E(x,w) = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M} e_{p,m}^2 \tag{6}$$

where x is the input vector; w is the weight vector; and $e_{p,m}$ is the training error at output m (from 1 to M) when applying pattern p (from 1 to P) and is defined as:

$$e_{p,m} = d_{p,m} - o_{p,m} \tag{7}$$

where d is the desired output vector; and o is the actual output vector. The parameter update rule of the gradient descent is performed using:

$$w_{k+1} = w_k - \alpha g_k \tag{8}$$

where k is the index of iterations; and α is the learning constant known as step size. Newton's method involves an expansion of the gradient using a Taylor series which results to an equation in matrix form as below:

$$\begin{bmatrix} -g_1 \\ -g_2 \\ \vdots \\ -g_N \end{bmatrix} = \begin{bmatrix} -\frac{\partial E}{\partial w_1} \\ -\frac{\partial E}{\partial w_2} \\ \vdots \\ -\frac{\partial E}{\partial w_N} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_N \partial w_N} & \frac{\partial^2 E}{\partial w_N \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_N \partial w_N} \end{bmatrix} \times \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \\ \vdots \\ \Delta w_N \end{bmatrix} \tag{9}$$

Gauss-Newton's algorithm proposed the Jacobian matrix to simplify the analysis as follows:

$$J = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_1} & \frac{\partial e_{1,1}}{\partial w_2} & \cdots & \frac{\partial e_{1,1}}{\partial w_N} \\ \frac{\partial e_{1,2}}{\partial w_1} & \frac{\partial e_{1,2}}{\partial w_2} & & \frac{\partial e_{1,2}}{\partial w_N} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e_{1,M}}{\partial w_1} & \frac{\partial e_{1,M}}{\partial w_2} & & \frac{\partial e_{1,M}}{\partial w_N} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e_{p,1}}{\partial w_1} & \frac{\partial e_{p,1}}{\partial w_2} & \ddots & \frac{\partial e_{p,1}}{\partial w_N} \\ \frac{\partial e_{p,2}}{\partial w_1} & \frac{\partial e_{p,2}}{\partial w_2} & & \frac{\partial e_{p,2}}{\partial w_N} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e_{p,M}}{\partial w_1} & \frac{\partial e_{p,M}}{\partial w_2} & & \frac{\partial e_{,M}}{\partial w_N} \end{bmatrix} \tag{10}$$

The gradient vector, g can also be written in terms of Jacobian matrix, J.

$$g = Je \tag{11}$$

where e is the error vector. Thus, the Gauss-Newton algorithm's parameter update rule is proposed as:

$$w_{k+1} = w_k - (J_k^T J_k)^{-1} J_k e_k \tag{12}$$

As a result, the parameter update rule of the Levenberg-Marquardt algorithm is given by:

$$w_{k+1} = w_k - (J_k^T J_k + \mu I)^{-1} J_k e_k \tag{13}$$

where μ is known as the combination coefficient and I is the identity matrix. The algorithm will employ

Gauss-Newton algorithm when μ is very small or nearly zero and will approximate to gradient descent method when μ is very large with $\alpha = 1/\mu$. Thus, throughout the training process, the Levenberg-Marquardt algorithm switches between gradient descent and Gauss-Newton's algorithms.

CONJUGATE GRADIENT METHOD

The conjugate gradient method is derived intermediately from gradient descent and Newton's method. Given gradient descent algorithm, the step size is determined via the exact line search procedure, which can significantly be slow in most real-world problems, although it is best known as the simplest gradient method (Mina & Mohammad-Mehdi 2018). The conjugate gradient method is faster in convergence, and it is simple as well as less complex. The derivation is presented below.

$$f(x) = \frac{1}{2}x^T Q_x - b^T x \tag{14}$$

where x has size n and Q is symmetric, positive definite and has size $n \times n$. At the point of zero gradients, the minimum of $f(x)$ is computed as:

$$\nabla f(x) = Qx - b = 0, Qx^* = \mathrm{b} \tag{15}$$

Let d_0, d_0, ..., d_0 be a finite set of vectors, and it is Q -orthogonal to each other if $d_i^T Q d_j = 0$ for all $i \neq j$. They are also linearly independent. Given a starting point x_0 and a Q-orthogonal set, the vector representing a move from x_0 to x^* is presented by:

$$x^* - x_0 = \sum_{i=0}^{n-1} \alpha_i d_i \tag{16}$$

where α_0, α_1, ...,α_{n-1} are scalars. The equation (16) is multiplied by $d_i^T Q$ and with $b = Qx$ will result in:

$$d_j^T (b - Qx_0) = \sum_{i=0}^{n-1} \alpha_i d_j^T Q\, d_i \tag{17}$$

Since the set of vectors d_i is Q-orthogonal,

$$\alpha_j = \frac{d_j^T (b - Qx_0)}{d_j^T Q d_j} \tag{18}$$

while $\nabla f(x_k)$ is denoted as g_k and $g_k = b\text{-}Qx_k$, it can be shown that:

$$\alpha_k = -\frac{d_k^T g_k}{d_k^T Q d_k} \tag{19}$$

If x_k is defined as:

$$x_k = x_0 + \sum_{j=0}^{k-1} \alpha_j d_j \tag{20}$$

then an iterative expression for x is

$$x_{k+1} = x_k + a_k d_k \tag{21}$$

Setting initial direction vector d_0 equals to the negative gradient at the initial point, $d_0 = -g_0$. The successive directions, d_{k+1} are a linear combination of the current gradient and the previous direction, and is given by:

$$d_{k+1} = -g_{k+1} + \beta_k d_k \tag{22}$$

The successive directions, d_{k+1} are Q-orthogonal, thus:

$$d_{k+1}^T Q d_k = [-g_{k+1} + \beta_k d_k]^T Q d_i = 0 \tag{23}$$

As a result,

$$\beta_k = \frac{g_{k+1}^T Q d_k}{d_k^T Q d_k} \tag{24}$$

The summary of Conjugate Gradient Method is as follows: 1. At $k = 1$, the initial point x_0 is selected. 2. $g_0 = \nabla f(x_0)$ if $g_0 = 0$, stop. Else set $d_0 = -g_0$. 3. $\alpha_k = -d_k^T g_k / d_k^T Q d_k$. 4. $x_{k+1} = x_k + a_k d_k$. 5. $g_{k+1} = \nabla f(x_{k+1})$, if $g_{k+1} = 0$, stop. 6. $\beta_k = g_{k+1}^T Q d_k / d_k^T Q d_k$. 7. $d_{k+1} = g_{k+1} + \beta_k d_k$. 8. If $k = n$-1, stop else $k = k$-1, go to step 3.

PERFORMANCE EVALUATION

The error function standard for minimization in neural networks is the sum of squared error. Considering that the appropriate error measures for forecasting has not reached a consensus as reported by Zhang (2012), the mean squared error, MSE, and the mean absolute percentage error, MAPE, are recommended as the most appropriate measures for forecasting.

$$MSE = \frac{1}{n}\sum_{t=1}^{n}\left(Y_t - \hat{Y}_t\right)^2 \tag{25}$$

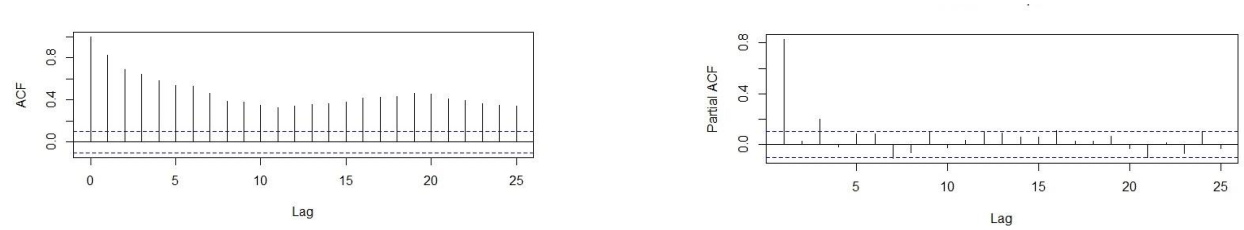
$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{\left|Y_t - \hat{Y}_t\right|}{Y_t} \times 100\%, Y_t \neq 0 \tag{26}$$

where Y_t is the observation at time t and $\hat{Y}_t$ is the predicted values.

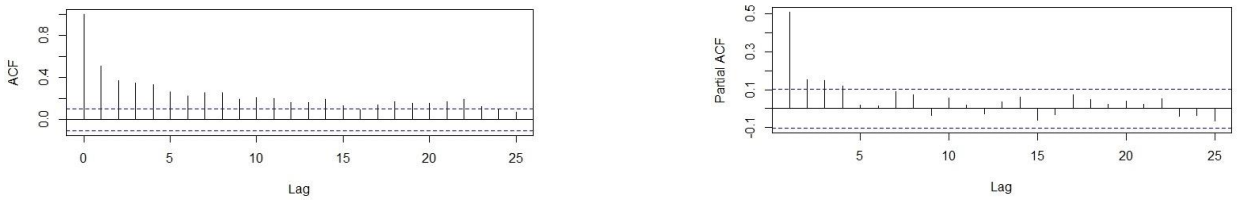
Results and Discussion

Among 11 air pollutants recorded from the continuous monitoring station, carbon monoxide (CO), ozone (O_3), particulate matter diameter 10 (PM_{10}) and Air Pollutant Index (API) were used in this study. CO, O_3 and PM_{10} were the three main pollutants that affecting Malaysia's air quality status. Meanwhile, the API was chosen in this study since the index is a simple and generalized method to assess the impact of air quality status on human health.

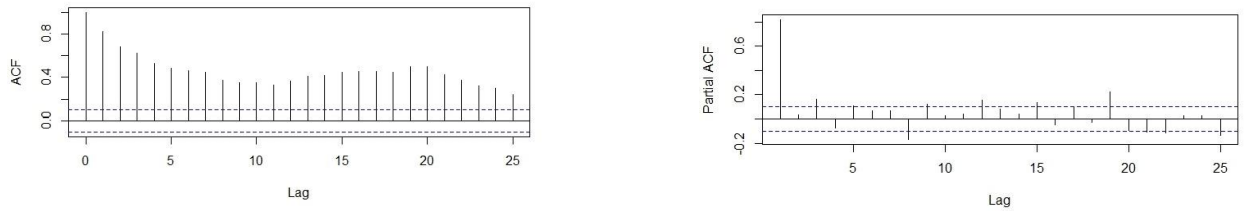
Figure 2(a) shows that the CO ACF dies down slowly as there were significantly large ACF values as the lag increases. A similar ACF pattern is found in Figure 2(c) for the PM_{10} pollutant. Thus, the result indicates both data were non-stationary. On the other hand, O_3 was stationary since, in Figure 2(b), the ACF dies down quickly as the lag increases. To model the data, stationary is required. Thus, the first level of differencing was used. The sample ACF and sample PACF of CO and PM_{10} after first differencing were shown in Figure 3.



(a) Carbon Monoxide (CO)



(b) Ozone (O_3)



(c) Particulate Matter diameter 10 (PM_{10})

FIGURE 2. ACF and PACF for CO, O_3 and PM_{10}

The values of p and q are limited up to order six as higher order models lead to higher values of BIC and restricted to the principle of parsimony; the smaller number of parameters is the best model. The best fitted model for CO was ARIMA (1,1,1), with the smallest BIC value of -121.45. For O_3, ARIMA (1,0,1) has the smallest value of BIC of -2582.72, and that makes the best fitted model for O_3 time series. Meanwhile, for PM_{10}, the best fitted model obtained is ARIMA (2,1,2) with BIC of 3382.4. Thus, all the three models can be written as follows:

i) *CO*

$$CO_t = 1.6928CO_{t-1} - 0.6928CO_{t-2} - 0.9468\varepsilon_{t-1} + \varepsilon_t$$

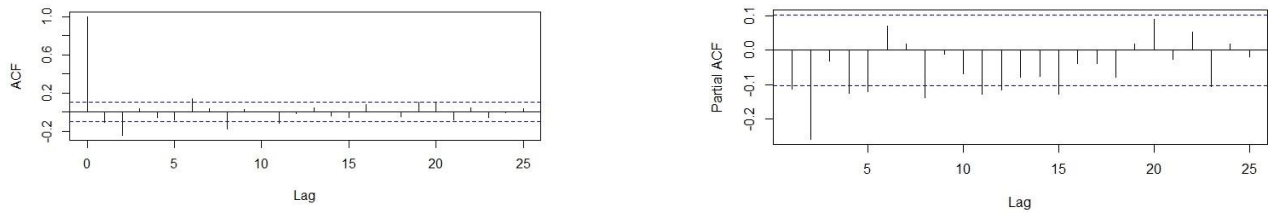
ii) O_3

$$O_{3t} = 0.8663O_{3t-1} - 0.5404\varepsilon_{t-1} + \varepsilon_t$$

iii) PM_{10}

$$PM_{10_t} = 0.7992PM_{10_{t-1}} + 0.7057PM_{10_{t-2}} - 0.5049PM_{10_{t-3}} + 0.0482\varepsilon_{t-1} - 0.8765\varepsilon_{t-1} + \varepsilon_t$$

The API in Malaysia was developed based on the API introduced by the United States Environmental Protection Agency (USEPA) and is determined by the calculation of sub-indexes of five main pollutants, namely



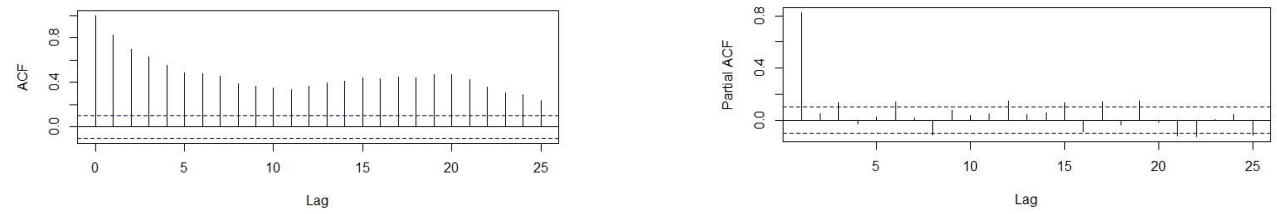
(a) Carbon Monoxide (CO)

(b) Particulate Matter diameter 10 (PM_{10})

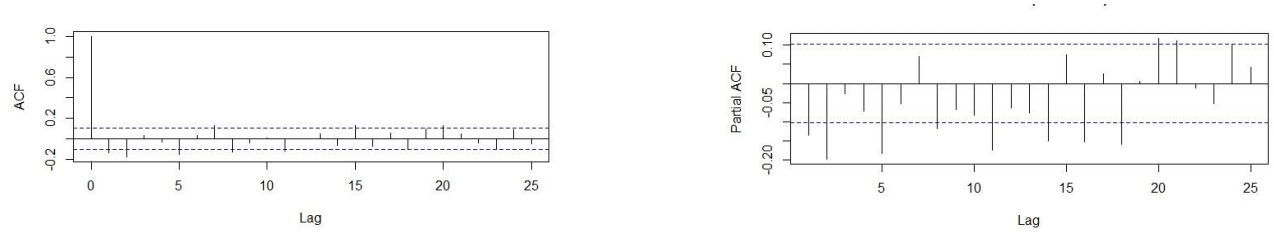
FIGURE 3. ACF and PACF for CO and PM_{10} after first differencing

PM_{10}, O_3, CO, sulfur dioxide (SO_2), and nitrogen dioxide (NO_2). Since PM_{10}, O_3 and CO had a tremendous effect on the API readings, these pollutants are also used in modelling API as the input in MLP. Figure 4(a) indicates that the API data was non-stationary since the ACF dies down slowly. Thus, first differencing was computed and the new ACF and PACF were given in Figure 4(b). Based on Figure 4(b), the best model for API is ARIMA (1,1,1) which gives the smallest BIC value of 2916.98. Therefore, the model can be written as follow:

$$API_t = 1.6682API_{t-1} - 0.6682API_{t-2} - 0.9188\varepsilon_{t-1} + \varepsilon_t$$



(a) Before differencing



(b) After Differencing

FIGURE 4. ACF and PACF for API before and after first differencing

TABLE 1. MLP inputs based on ARIMA modelling

Type	Variables	Lagged variables
Explanatory variables	CO	CO_t, CO_{t-1}
	O3	O_{3_t}, $O_{3_{t-1}}$
	PM10	PM_{10_t}, $PM_{10_{t-1}}$, $PM_{10_{t-2}}$
Response variables	API	API_t

The MLP models was programmed in MATLAB Software. The selected input is based on ARIMA modelling given in Table 1. Data pre-processing was also carried out to describe the important relationships and to set up more uniform data in neural network training. All the data will undergo pre-processing by z-score normalization, where it uses the mean and standard deviation to transform each feature to have zero mean and unit variance. The advantage of z-score normalization is the effects of outliers in the data does not diminish. The data from January 2015 to Dec 2015 was split into 70% training, 15% validation, and 15% testing. On the other hand, the data in January 2016 was used to evaluate the ability of the model to forecast future values.

Seven lagged variables of CO, O_3 and PM_{10} which include CO_t, CO_{t-1}, O_{3_t}, $O_{3_{t-1}}$, PM_{10_t}, $PM_{10_{t-1}}$, and $PM_{10_{t-2}}$ were selected as MLP inputs while API_t served as MLP target. Seven inputs were used with different architectures under the two training algorithms: Levenberg-Marquardt and the conjugate gradient method. A single hidden layer with hidden neurons used was between two and ten. All the models were repeated 50 times with different initial values of weights, and the best model with the lowest value of MSE was taken. The result is shown in Table 2.

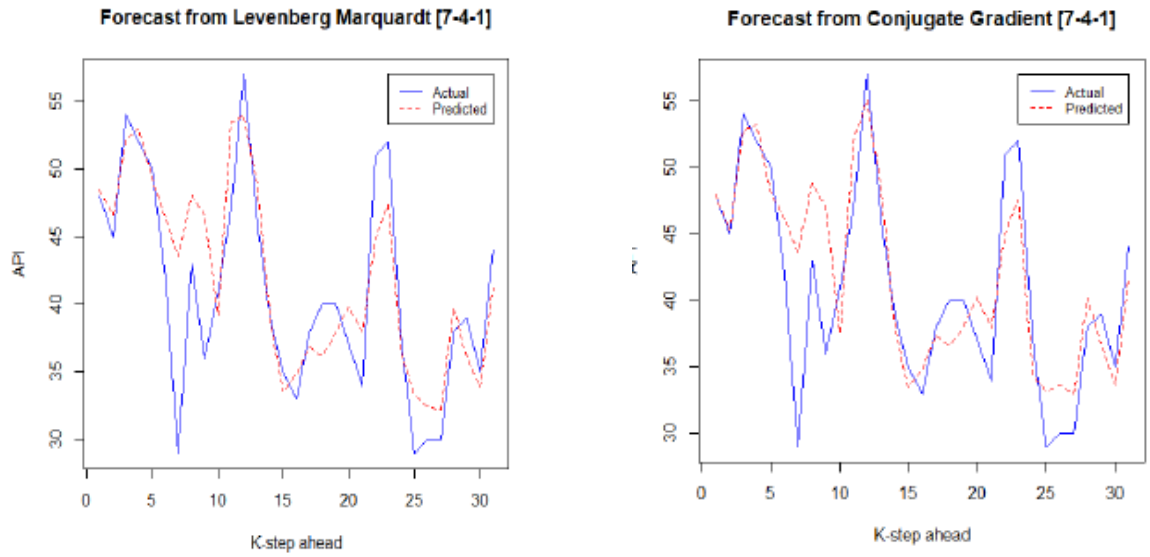
TABLE 2. Comparison of MLP models with different architecture

Algorithm	Model	Architecture	Performance	
			MSE	MAPE
Levenberg-Marquardt	1	[7-2-1]	18.3174	4.4262
	2	[7-4-1]	**16.4634**	4.2534
	3	[7-6-1]	16.5208	**4.1746**
	4	[7-8-1]	18.4410	4.4784
	5	[7-10-1]	19.1623	4.2680
Conjugate gradient	6	[7-2-1]	18.9717	4.5014
	7	[7-4-1]	**16.7298**	4.6122
	8	[7-6-1]	18.4864	5.1317
	9	[7-8-1]	17.7246	4.9169
	10	[7-10-1]	17.6003	**4.4648**

TABLE 3. API forecasting performance

Model		MSE	MAPE
ARIMA		96.7395	23.4375
MLP			
-	LevenbergMarquardt	**19.4273**	**8.5688**
-	Conjugate Gradient	19.6007	8.8220

Based on Table 2, the Levenberg-Marquardt performed well under [7-4-1] architecture as it had the lowest MSE value of 16.4634 compared to other models. Similarly, the conjugate gradient method had its best performance under [7-4-1] architecture with MSE of 16.7298. The lowest MAPE value for Levenberg-Marquardt and the conjugate gradient method was 4.2534 and 4.6122, respectively. Since the consistency of the result obtained in [7-4-1] architecture, the developed models are then further used in forecasting, where the forecasting result is shown in Table 3.



(a) Levenberg-Marquardt (b) Conjugate Gradient

FIGURE 5. Actual and forecast by Levenberg-Marquardt and conjugate gradient

Based on Table 3, modelling the API data with ARIMA (1,1,1) model has relatively high MSE and MAPE value for forecast error even though it is the best fitted model. The forecast error based on MAPE value indicates that the forecast is off by 23.4375%. For MLP modelling, it was found out that the well-performed model using the Levenberg-Marquardt algorithm compared to the conjugate gradient method with its MSE value slightly lower. Similarly, in MAPE result. The forecast error was 8.5688% which mean as a highly accurate forecast, given that the MAPE is less than 10%. This is because the Levenberg-Marquardt is competent in training small and medium-sized networks (Yu & Wilamowski 2011). The actual values and forecast values from both Levenberg-Marquardt and the conjugate gradient method model were plotted, respectively, as shown in Figure 5.

CONCLUSION

The air quality data from Putrajaya monitoring station in the year 2015 was used to forecast API using ARIMA and MLP models. Using classical Box-Jenkins methodology, the best parsimonious model suggested for predicting API future values was ARIMA (1,1,1) given its lowest BIC value. The classical ARIMA and MLP models were also compared to investigate their performance in prediction. MLP models were recommended over ARIMA models for forecasting as the results confirmed that forecast values by MLP models close to the actual values.

On the other hand, various architectures of the MLP in terms of number of inputs, number of hidden neurons and training algorithm were used for fitting of the data as well as forecasting. It is essential to consider different MLP models such as the architectures, activation function, and initial weights to obtain a good fitting model that will converge and produce sensible predictions. When two inputs were used in MLP, the Levenberg-Marquardt algorithm had better performance than the conjugate gradient method. Levenberg-Marquardt model was more suitable to build the forecast model with its small MSE and MAPE values. The MLP models were also fitted by using seven inputs. It is shown

that the training algorithm that best minimizes the error was the Levenberg-Marquardt algorithm.

There is plenty of scopes that can be conducted by using ANN or MLP. One of the recommendations for future work is to study on other configuration parameters of the algorithms concerning the forecast accuracy. The choices of MLP architectures, training algorithms, activation function and appropriate configuration parameters must be selected carefully and greatly to produce the best fit model.

ACKNOWLEDGEMENTS

The authors would like to express our gratitude to the editor and referees for their valuable comments and to the Department of Environment (DOE) Malaysia for providing the pollutants data.

REFERENCES

An, N.H. & Anh, D.T. 2015. Comparison of strategies for multi-step-ahead prediction of time series using neural network. *2015 International Conference on Advanced Computing and Applications (ACOMP)*. pp. 142-149.

Chen, X., Wang, G., Zhou, W., Chang, S. & Sun, S. 2006. Efficient Sigmoid function for neural networks based FPGA design. *International Conference on Intelligent Computing*. pp. 672-677.

De Cosmi, V., Mazzocchi, A., Milani, G.P., Calderini, E., Scaglioni, S., Bettocchi, S., D'Oria, V., Langer, T., Spolidoro, G.C.I., Leone, L., Battezzati, A., Bertoli, S., Leone, A., De Amicis, R.S., Foppiani, A., Agostoni, C. & Grossi, E. 2020. Prediction of resting energy expenditure in children: May artificial neural networks improve our accuracy? *Journal of Clinical Medicine* 9(4): 1026.

Dong, G., Fataliyev, K. & Wang, L. 2013. One-Step and multi-step ahead stock prediction using backpropagation neural networks. *2013 9th International Conference on Information, Communications & Signal Processing*. pp. 1-5.

Evans, N.J. 2019. Assessing the practical differences between model selection methods in inferences about choice response time tasks. *Psychonomic Bulletin & Review* 26(4): 1070-1098.

Jain, A.K., Mao, J. & Mohiuddin, K.M. 1996. Artificial Neural Networks: A tutorial. *Computer* 29(3): 31-44.

Kaastra, I. & Boyd, M. 1996. Designing a Neural Network for forecasting financial and economic time series. *Neurocomputing* 10(3): 215-236.

Kavzoglu, T. 1999. Determining optimum structure for Artificial Neural Networks. *Proceedings of the 25th Annual Technical Conference and Exhibition of the Remote Sensing Society,* Cardiff, UK. 8-10 September. pp. 675-682.

Kermani, B.G., Schiffman, S.S. & Nagle, H.T. 2005. Performance of the Levenberg–Marquardt neural network training method in electronic nose applications. *Sensors and Actuators B: Chemical* 110(1): 13-22.

Mahmoud Khaki, Ismail Yusoff, Nur Fadilah Islami & Nur Hayati Hussin. 2016. Artificial Neural network technique for modeling of groundwater level in Langat Basin, Malaysia. *Sains Malaysiana* 45(1): 19-28.

Lodwich, A., Rangoni, Y. & Breuel, T. 2009. Evaluation of robustness and performance of early stopping rules with multi layer perceptrons. *2009 International Joint Conference on Neural Networks*. pp. 1877-1884.

Nurulilyana Sansuddin, Nor Azam Ramli, Ahmad Shukri Yahaya, Noor Faizah Fitri Md Yusof, Nurul Adyani Ghazali & Wesam Ahmed Al Madhoun. 2011. Statistical analysis of PM_{10} concentrations at different locations in Malaysia. *Environmental Monitoring and Assessment* 180(1-4): 573-588.

Saratha Sathasivam, Mohd. Asyraf Mansor, Ahmad Ismail, Siti Jamaludin, Mohd Kasihmuddin, & Mustafa Mamat. 2020. Novel random k satisfiability for $k \leq 2$ in Hopfield neural network. *Sains Malaysiana* 49(11): 2847-2857.

Sathya, R. & Annamma Abraham. 2013. Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence* 2 (February). https://dx.doi.org/10.14569/IJARAI.2013.020206

Sulafa Hag Elsafi. 2014. Artificial Neural Networks (ANNs) for flood forecasting at Dongola Station in the River Nile, Sudan. *Alexandria Engineering Journal* 53(3): 655-662.

Sheela Tiwari, Ram Naresh & Rahul Jha. 2013. Comparative study of backpropagation algorithms in neural network based identification of power system. *International Journal of Computer Science and Information Technology* 5(August): 93-107.

Mina Torabi & Mohammad-Mehdi Hosseini. 2018. A new descent algorithm using the three-step discretization method for solving unconstrained optimization problems. *Mathematics* 6 (April): 63.

Yu, H. & Wilamowski, B.M. 2011. Lavenberg-Marquardt Training. In *The Industrial Electronics Handbook: Intelligent Systems*, 2nd ed. Chapter 12, edited by Wilamowski, B.M. & Irwin, J.D. Boca Raton: CRC Press. pp. 12-1 to 12-15.

Zafer Cömert & Adnan Fatih Kocamaz. 2017. A Study of artificial neural network training algorithms for classification of cardiotocography signals. *Bitlis Eren University Journal of Science and Technology* 7(2): 93-103.

Zhang, G.P. 2012. Neural networks for time-series forecasting BT. In *Handbook of Natural Computing*, edited by Rozenberg, G., Bäck, T. & Kok, J.N. Berlin, Heidelberg: Springer Berlin Heidelberg. pp. 461-477.

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

Zhu, L., Lim, C. & Zhang, J. 2020. Prediction of risk: Decoding the serial dependence of stock return volatility with Copula. *Journal of Hospitality & Tourism Research* 45(1): 6-27.

*Corresponding author; email: nurhaizum_ar@upm.edu.my