Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 9 (6): 1001-1013, 2009
ISSN 1812-5654

Application of Artificial Neural Networks for Airline Number of Passenger Estimation in Time Series State

[1]M. Zandieh, [2]A. Azadeh, [3]B. Hadadi and [4]M. Saberi
[1]Department of Industrial Management, Faculty of Management and Accounting,
Shahid Beheshti University, G.C., Tehran, Iran
[2]Department of Industrial Engineering, Faculty of Engineering, University of Tehran, Tehran, Iran
[3]Department of Industrial Engineering, Faculty of Mechanical and Industrial Engineering,
Qazvin Azad University of Ghazvin, Iran
[4]Department of Industrial Engineering, University of Tafresh, Tafresh, Iran

Abstract: This study presents an integrated Artificial Neural Networks (ANN) to estimate and predict airline number of passenger in Iran. All type of ANN-Multi Layer Perceptron (MLP) is examined to this estimation. The ANN models are implemented on MATLAB software. Auto-Correlation Function (ACF) is utilized to define input variables. Finally, the best type of ANN-MLP is determined with Data Envelopment Analysis (DEA). Kruskal-Wallis test is used for asses the impact of raw data, preprocessed data and post process method on ANN performance. Monthly airline number of passenger of Iran airline from 1993 to 2005 is considered as the case of this study.

Key words: Prediction, artificial neural network, data envelopment analysis, airline number of passenger, pre-processed data

INTRODUCTION

There have been many studies on ANN models in different cases. An ANN is configured for a specific application, such as pattern recognition, function approximation, data classification and so on in different areas of science. Time series modeling is one of the main applications. Many researchers showed ANN's comparability and superiority to conventional methods for estimating functions (Hill *et al*., 1996; Kohzadi *et al*., 1996; Stern, 1996; Chiang *et al*., 1996; Indro *et al*., 1999; Hwarng, 2001; Azadeh *et al*., 2007a,b).

The main aim of this research is to combine conventional time series concepts with ANN. We show these concepts are more useful in improving ANN performance. These concepts are preprocessing (for madding process, covariance stationary), post processing (to access main data) and principle component analysis (for input selection). Exploring the literature reveals that combination of traditional concepts with ANN to model time series has been rarely done. Although data preprocessing concept is considered in some literature, but the covariance stationary concept in data preprocessing is ignored (Nayak *et al*., 2004; Karunasinghe and Liong, 2006; Tseng *et al*., 2002; Niska *et al*., 2004; Aznarte *et al*., 2007; Gareta *et al*., 2006; Jain and Kumar, 2007).

ARTIFICIAL NEURAL NETWORKS

Artificial neural networks are a promising alternative to econometric models. An ANN is an information processing paradigm that is inspired by the biological nervous systems, such as the brain, process information. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition, function approximation or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true for ANNs as well. They are made up simple processing units which are linked by weighted connections to form structures that are able to learn relationships between sets of variables. This heuristic method can be useful for nonlinear processes that have unknown functional forms (Enders, 2004). There have been many studies of ANN models in time series

Corresponding Author: M. Zandieh, Department of Industrial Management, Faculty of Management and Accounting, Shahid Beheshti University, Tehran, Iran

forecasting (Hill *et al*., 1996; Kohzadi *et al*., 1996; Hawrng, 2001; Tseng *et al*., 2002; Niska *et al*., 2004; Jain and Kumar, 2007; Palmer *et al*., 2006; Zhang, 2001), but traditional concepts such as covariance stationary and input selection are ignored. In this study, we show that using these concepts improves the performance of ANN model. Then, the best ANN architecture is selected for airline number of passenger consumption estimation.

Among the different networks, the feed forward neural networks or Multi Layer Perceptron (MLP) are the most commonly used in engineering. MLP networks are normally arranged in three layers of neurons, the input layer and output layer represent the input and output variables of the model and between them lay one or more hidden layers which hold the networks ability to learn non-linear relationships.

Architecture selection is one major issue with implications on the empirical results and consists of:

- Input and output variables number
- Hidden layers' number
- Hidden and output activation functions
- Learning algorithm

All of the above issues are open questions today and there are several answers to each one. The hidden units number is determined by a trial-error process. Few neurons in hidden layers (hidden units) can lead to under fitting. However, too many neurons can cause over fitting. The actual number of neurons required in the hidden layer must be found by trial and error. Moreover, the inputs are used by the network must be effective on the value of output(s), in fact the input and output variables should be identified carefully, because enable the network to learn relationships quicker and use fewer hidden units.

All networks have a single hidden layer because the single hidden layer network is found to be sufficient to model any function (Cybenko, 1989). To find the appropriate number of hidden nodes, networks with one to q nodes in their hidden layer are constructed. The value of q is optional and should be changed if the goal error has not met.

Learning algorithms: Famous type of different back-propagation algorithms is explained in following.

Batch training with weight and bias learning rules (Trainb): Trainb trains a network with weight and bias learning rules with batch updates. The weights and biases are updated at the end of an entire pass through the input data.

BFGS quasi-Newton back-propagation (Trainbfg): Trainbfg is a network training function that updates weight and bias values according to the BFGS quasi-Newton method.

Bayesian regularization back-propagation (Trainbr): Trainbr is a network training function that updates the weight and bias values according to Levenberg-Marquardt optimization. It minimizes a combination of squared errors and weights and then determines the correct combination so as to produce a network that generalizes well. The process is called Bayesian regularization (Powell, 1977).

Cyclical order incremental training with learning functions (Trainc): Trainc trains a network with weight and bias learning rules with incremental updates after each presentation of an input. Inputs are presented in cyclic order.

Conjugate gradient back propagation with Powell-Beale restarts (Traincgb): Traincgb is a network training function that updates weight and bias values according to the conjugate gradient back propagation with Powell-Beale restarts (Powell, 1977).

Conjugate gradient back-propagation with Fletcher-Reeves updates (Traincgf): Traincgf is a network training function that updates weight and bias values according to the conjugate gradient back-propagation with Fletcher-Reeves updates.

Conjugate gradient back-propagation with Polak-Ribiere updates (Traincgp): Traincgp is a network training function that updates weight and bias values according to the conjugate gradient back-propagation with Polak-Ribiere updates.

Gradient descent back-propagation (Traingd): Traingd is a network training function that updates weight and bias values according to gradient descent.

Gradient descent with adaptive learning rate back-propagation (Traingda): Traingda is a network training

function that updates weight and bias values according to gradient descent with adaptive learning rate.

Gradient descent with momentum back-propagation (Traingdm): Traingdm is a network training function that updates weight and bias values according to gradient descent with momentum.

Gradient descent with momentum and adaptive learning rate back-propagation (Traingdx): Traingdx is a network training function that updates weight and bias values according to gradient descent momentum and an adaptive learning rate.

Levenberg-Marquardt back-propagation (Trainlm): Trainlm is a network training function that updates weight and bias values according to Levenberg-Marquardt optimization.

One step secant back-propagation (Trainoss): Trainoss is a network training function that updates weight and bias values according to the one step secant method.

Random order incremental training with learning functions (Trainr): For each epoch, all training vectors (or sequences) are each presented once in a different random order, with the network and weight and bias values updated after each individual presentation.

Resilient back-propagation (Trainrp): Trainrp is a network training function that updates weight and bias values according to the resilient back-propagation algorithm (Rprop).

Sequential order incremental training with learning functions (Trains): Trains trains a network with weight and bias learning rules with sequential updates. The sequence of inputs is presented to the network with updates occurring after each time step.

Scaled conjugate gradient back-propagation (Trainscg): Trainscg is a network training function that updates weight and bias values according to the scaled conjugate gradient method.

Cross Validation Test Technique (CVTT): In CVTT the data set is first split into several parts. Then, one part is utilized for testing and the rest are saved for training purpose. These steps are repeated until all parts used as testing set. The final product of CVTT is the mean accuracy of total runs.

DATA PRE-PROCESSING

Covariance stationary is one of the basic assumptions in time-series and should be studied for the models. Moreover, using preprocessed data is more useful in most heuristic methods (Zhang and Qi, 2005). If the models are not covariance stationary, the most suitable preprocessed method should be defined and applied.

The first Difference method: As mentioned, the first step in the Box-Jenkins method is to transform the data so as to make it stationary. The difference method was proposed by Box and Jenkins (1976).

Tseng *et al.* (2002) also used this method in their article in which the estimation of time series functions using heuristic approach is done. In this method, the following transformation should be applied:

$$y_t = x_t - x_{t-1} \quad (1)$$

With a bit change in 1 the first difference of the logarithm is:

$$y_t = \log (x_t) - \log (x_{t-1}) \quad (2)$$

RESEARCH METHODOLOGY

This algorithm has the following basic steps

Step 1: Divide data into two sets, one for estimating the models called train data set and the other one for evaluating the validity of the estimated model called test data set. Usually, train data set contains 70 or 90% of all data and the remaining data are used in test data set (Aznarte *et al.*, 2007).

Step 2: The stationary assumption should be studied for all type of ANN model. If the process is not covariance stationary, the most suitable preprocessing method should be selected and applied to the model.

Step 3: Determination of input variables for each model should be done. input variables can be selected using autocorrelation function (ACF) in most heuristic methods, selection of input variables is experimental or

based on the trial and error method (Hwarng *et al*., 2001; Nayak *et al*., 2004; Karunasinghe *et al*., 2006; Tseng *et al*., 2002; Zhang and Qi, 2005; Zhang and Hu, 1998; Palmer *et al*., 2006; Kim *et al*., 2004; Zhang, 2001).

Step 4: Running and estimating of all models are done in this step.

The plausible architecture of ANN is constructed in this step.

Step 5: The predicting ability for each model is evaluated in this step. Furthermore, the comparison of each model with actual data is done and error of each model is calculated. MAPE is used to compare the models.

CASE STUDY

The proposed algorithm is applied to 130 data, which are the monthly consumption values from April 1993 to February 2005.

Step 1: All of 129 pre-processing data divided into 117 training data and 12 test data.

Step 2: It can be shown in Fig. 1a in that raw data have a trend. As removing the trend is needed for more precise estimation and also for studying the impact of preprocessing on ANN, preprocessing methods are applied on data. Finally, the best preprocessing method for our data which can make the model covariance stationary is selected. The results of applying preprocessing methods for given data are as follows:

The first difference method: The preprocessed data using this method are shown in Fig. 1b. Although, the first difference of the series seems to have a constant mean, the variance is an increasing function of time. So, this method is not covariance stationary and can not be used for data preprocessing.

It can be shown in Fig. 1c, which shows the first difference of the logarithm of consumption data, that this

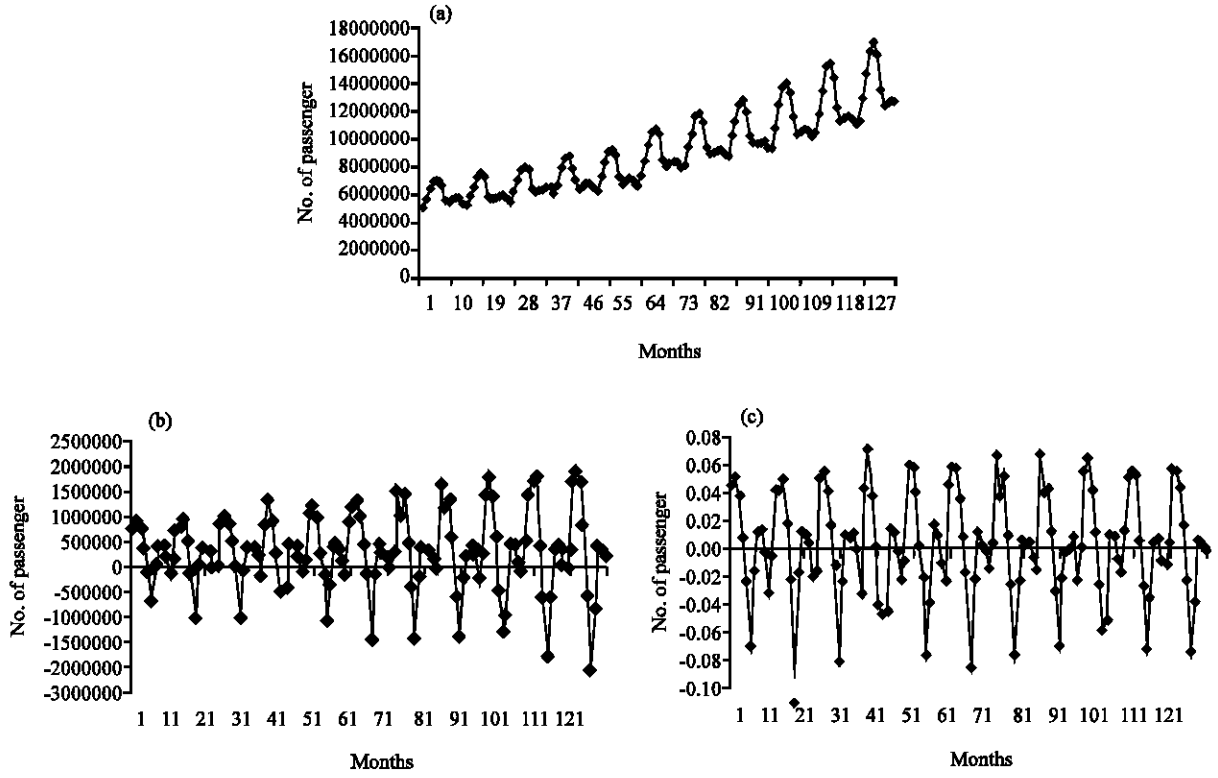

Fig. 1: (a) Raw data, (b) Proposed data by first differences method and (c) Proposed data by first differences of the Logarithm methods

method is the most likely candidate to be covariance stationary. So, it is the most applicable preprocessing method for our data.

Step 3: For all ANN models, ACF approach is used to select input variables. Figure 2 shows that y_t is the function of consumption in the 1st, 11th and 12th lags in preprocessed data.

Step 4: (Estimation of airline number of passenger by ANN-MLP): In order to get the best ANN, 17 MLP models are tested to find the best architecture. Three parts of train data are constructed for test in different running; the final product of CVTT is the mean accuracy of total runs. Table 1 shows the MAPE value of 340 ANN-MLP models. Truly, for each learning algorithm, networks with having 1 to 20 nodes is constructed and implemented. Best architectures of train algorithms are shown in Fig. 3. In Table 2 statistic variable related to Table 1 is showed.

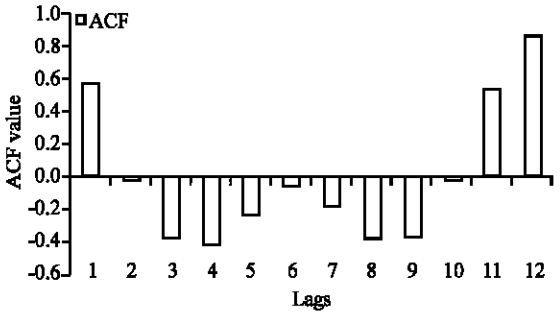

Fig. 2: ACF charts for select input variable

Table 1: MAPE value of learning algorithms in ANN-MLP mode

	No. of nodes in hidden layer 1									
Learning algorithm	1	2	3	4	5	6	7	8	9	10
Trainb	0.208	0.154	0.627	1.122	1.062	0.729	7.799	2.838	0.965	2.039
Trainbfg	0.05	0.06	0.062	0.062	0.072	0.065	0.067	0.067	0.059	0.104
Trainbr	0.023	0.021	0.021	0.022	0.022	0.022	0.023	0.022	0.022	0.019
Trainc	0.041	0.053	0.040	0.045	0.049	0.027	0.051	0.052	0.048	0.051
Traincgb	0.056	0.062	0.045	0.035	0.052	0.043	0.052	0.061	0.111	0.106
Traincgf	0.063	0.062	0.024	0.042	0.039	0.086	0.026	0.043	0.075	0.038
Traincgp	0.052	0.053	0.053	0.039	0.064	0.045	0.043	0.116	0.047	0.095
Traingd	0.508	0.160	0.281	0.427	0.516	0.600	0.661	0.609	0.564	0.543
Traingda	0.053	0.131	0.132	1.185	0.170	1.420	0.265	0.610	0.283	0.243
Traingdm	0.349	0.176	0.645	1.160	1.236	1.007	9.602	0.527	1.182	9.230
Traingdx	0.155	0.159	0.377	2.876	0.454	5.706	2.723	0.182	0.415	2.498
Trainlm	0.026	0.041	0.028	0.019	0.023	0.023	0.024	0.024	0.026	0.022
Trainoss	0.048	0.040	0.045	0.043	0.048	0.051	0.050	0.035	0.204	0.090
Trainr	0.045	0.050	0.041	0.028	0.053	0.028	0.040	0.049	0.039	0.090
Trainrp	0.039	0.053	0.052	0.107	0.190	0.097	0.389	0.122	0.159	0.420
Trains	4.168	5.751	32.94	42.97	18.12	24.64	2.588	34.46	0.933	117.3
Trainscg	0.070	0.062	0.040	0.026	0.029	0.045	0.078	0.061	0.042	0.331
	No. of nodes in hidden layer 1									
Learning algorithm	11	12	13	14	15	16	17	18	19	20
Trainb	1.653	4.517	2.647	1.599	34.21	226.8	11.53	3.231	0.950	11.69
Trainbfg	0.101	0.103	0.075	0.100	0.130	0.113	0.102	0.063	0.093	0.249
Trainbr	0.023	0.022	0.021	0.027	0.020	0.024	0.022	0.024	0.019	0.024
Trainc	0.056	0.048	0.062	0.040	0.063	0.064	0.086	0.065	0.070	0.087
Traincgb	0.049	0.075	0.138	0.088	0.061	0.092	0.131	0.150	0.176	0.257
Traincgf	0.156	0.075	0.072	0.153	0.110	0.215	0.125	0.261	0.525	0.129
Traincgp	0.099	0.092	0.054	0.145	0.289	0.196	0.237	0.130	0.151	0.275
Traingd	0.900	0.823	0.650	0.939	5.027	0.948	18.66	1.173	1.518	1.413
Traingda	0.191	0.141	0.393	0.342	0.323	0.642	0.421	0.233	0.299	0.645
Traingdm	1.867	2.002	6.393	4.662	3.023	1.632	42.83	6.143	2.985	333.1
Traingdx	1.019	0.306	0.594	0.503	0.661	0.450	0.727	2.856	0.742	15.76
Trainlm	0.027	0.021	0.025	0.024	0.023	0.019	0.024	0.024	0.026	0.028
Trainoss	0.072	1.523	0.199	0.184	0.084	0.355	0.206	0.344	0.146	0.228
Trainr	0.078	0.048	0.059	0.084	0.051	0.094	0.085	0.082	0.081	0.066
Trainrp	0.438	0.147	0.203	0.569	0.772	0.320	0.240	0.608	1.076	0.291
Trains	2.406	125.7	354.431	8.358	1622	148.1	2189	3479.6	38.614	2912
Trainscg	0.043	0.217	0.113	0.192	0.275	0.170	0.093	0.221	0.102	253.3

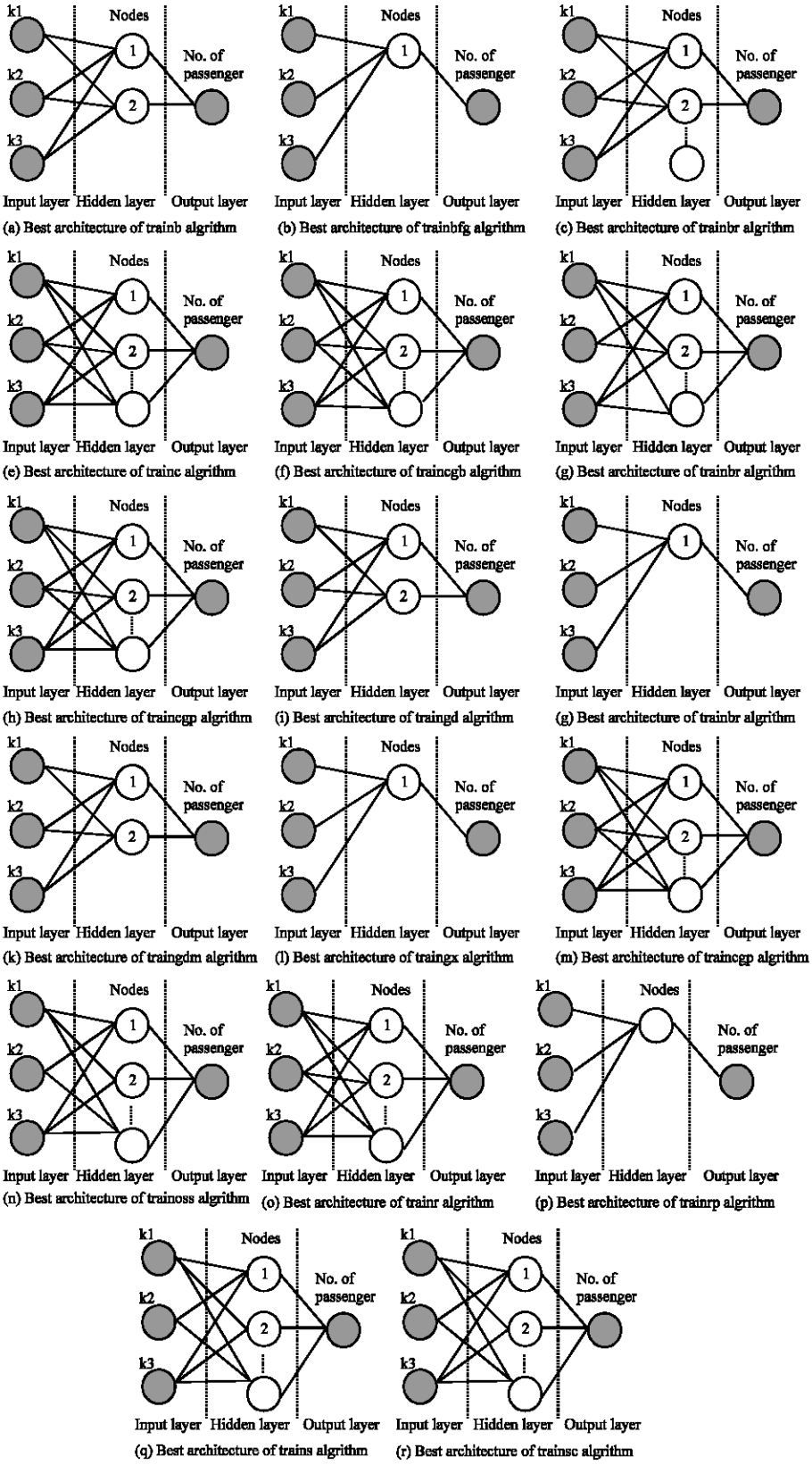

Fig. 3: The best architecture of the selected ANN-MLP Learning algorithm

Table 2: Statistical variable related to ANN-MLP models

Learning algorithm	AVEMAPE		MinMAPE		MaxMAPE		StdMAPE	
	With*	Without*	With	Without	With	Without	With	Without
Trainb	15.823	1.729	0.154	0.403	226.872	5.855	50.276	0.995
Trainbfg	0.091	0.148	0.058	0.081	0.249	0.255	0.043	0.100
Trainbr	0.022	0.038	0.019	0.025	0.027	0.075	0.002	0.020
Trainc	0.055	0.395	0.027	0.356	0.087	0.535	0.015	0.528
Traincgb	0.092	0.185	0.035	0.063	0.257	0.528	0.056	0.130
Traincgf	0.116	0.228	0.024	0.074	0.525	0.800	0.115	0.194
Traincgp	0.114	0.321	0.039	0.056	0.289	1.913	0.079	0.337
Traingd	1.846	1.529	0.160	0.243	18.661	5.030	4.086	1.596
Traingda	0.406	0.779	0.053	0.090	1.420	2.639	0.352	0.678
Traingdm	21.489	1.334	0.176	0.335	333.130	4.660	73.944	0.948
Traingdx	1.958	1.129	0.155	0.065	15.765	3.829	3.543	0.898
Trainlm	0.025	0.053	0.019	0.029	0.041	0.087	0.004	0.024
Trainoss	0.200	0.417	0.035	0.065	1.523	2.819	0.197	0.447
Trainr	0.060	0.692	0.028	0.678	0.094	0.722	0.022	0.534
Trainrp	0.315	0.435	0.039	0.112	1.076	0.867	0.263	0.326
Trains	558.261	2.632	0.933	0.694	3479.637	5.170	765.985	1.554
Trainscg	12.776	0.236	0.026	0.073	253.317	0.856	22.004	0.157

RESULT ANALYSIS

Comparison of ANN models by data envelopment analysis: DEA is a non-parametric method that uses linear programming to calculate the efficiency in a given set of Decision-Making Units (DMUs) (Charnes *et al*., 1978; Zhu, 1998). The DMUs that make up a frontier envelop, the less efficient firms and the relative efficiency of the firms is calculated in terms of scores on a scale of 0 to 1, with the frontier firms receiving a score of 1. DEA models can be input or output oriented and can be specified as constant returns to scale (CRS) or variable returns to scale (VRS).

We have utilized DEA method to comprise ANN models. Since MAPE alone may not be appropriate in comparison process thus we need to utilize other variables, involving AVEMAPE, MinMAPE, MaxMAPE and StdMAPE for each ANN models.

Where:

- **Variable MAPEjk is defined as:**
  Mean Absolute Percentage Error of ANN, with regard to kth part of data, used as test data and j node in hidden layer.
- **Variable AVEMAPE$_j$ is defined as:**
  AVEMAPE$_j$ = Average {MAPE$_{jk}$: k = 1, 2...} or average of MAPE in all constructed ANN with regard to j node in hidden layer.
- **AVEMAPE is defined as:**
  AVE = Average (AVEMAPE$_j$)
- **StdMAPE is defined as:**
  StdMAPE = std (AVEMAPE$_j$)
- **MaxMAPE is defined as:**
  MaxMAPE = Max (AVEMAPE$_j$)

  MinMAPE is defined as:
  MinMAPE = Min (AVEMAPE$_j$)

Error variables are shown in Table 2. DEA method will effectively take into account the values of these variables to Study the performance of ANN models. We treated value of MAPE, MIN, MAX and STD for each model as inputs with the specific output of 1 of each reduction. Calculated full rank efficiency scores along with ANN performance ranks are shown in Table 3. Examination of Table 3 shows that ANNs with Levenberg-Marquardt back-propagation and Bayesian regularization back-propagation algorithms and preprocessed data are the best. Also, examination of Table 3 shows that 12 of initial 17 learning algorithm use pre process and post process method. So five learning algorithm have good performance even without using preprocessed data. These are: Trainbr, trainlm, traingdx, traincgp and traincgb.

Comparison of ANN models by Kruskal-Wallis test: Here, Kruskal-Wallis test as a strong statistical tool is used in comparison process again. Failing the hypothesis of normality (non sufficient number of data) is the reason of using Kruskal-Wallis test instead of parametric version of ANOVA. The assumption behind this test is that the measurements come from a continuous distribution, but not necessarily a normal distribution. The test is based on an analysis of variance using the ranks of the data values.

Examinations of Fig. 4-6 shows post processed data have positive affect on ANN performance.

Examinations of Fig. 7 shows post process of ANN result after preprocess is vital.

Necessity to Use Post process method: Table 4 and Fig. 8 shows the Output of one ANN that uses preprocessed data. Examination Table 4 and the Fig. 8 shows that ANN model had acceptable performance but if the out of ANN

Table 3: Full rank efficiency of ANN-MLP model

Learning algorithm	Learning algorithm	ANN model efficiency scores	ANN model efficiency scores
Trainbrw	Trainosswo	0.292307692	1.518518519
Trainlmw	Trainscgwo	0.260273973	1.0
Traincgfw	Traincgfwo	0.256756757	0.791666667
Trainbrwo	Trainbfgwo	0.234567901	0.76
Trainscgw	Traingdawo	0.211111111	0.730769231
Traincw	Trainrpwo	0.169642857	0.703703704
Trainrw	Trainbw	0.123376623	0.678571429
Trainlmwo	Traingdxw	0.122580645	0.655172414
Traincgpw	Traingdw	0.11875	0.542857143
Trainossw	Traingdmw	0.107954545	0.542857143
Traincgpw	Traingdwo	0.0781893	0.487179487
Trainrpw	traingdmwo	0.056716418	0.487179487
Traingdaw	Traincwo	0.055696203	0.358490566
Traincgpwo	Trainbwo	0.047146402	0.339285714
Trainbfgw	Trainrwo	0.037396122	0.327586207
Traincgpwo	Trainswo	0.027377522	0.301587302
Traingdxwo	Trainsw	0.020364416	0.292307692

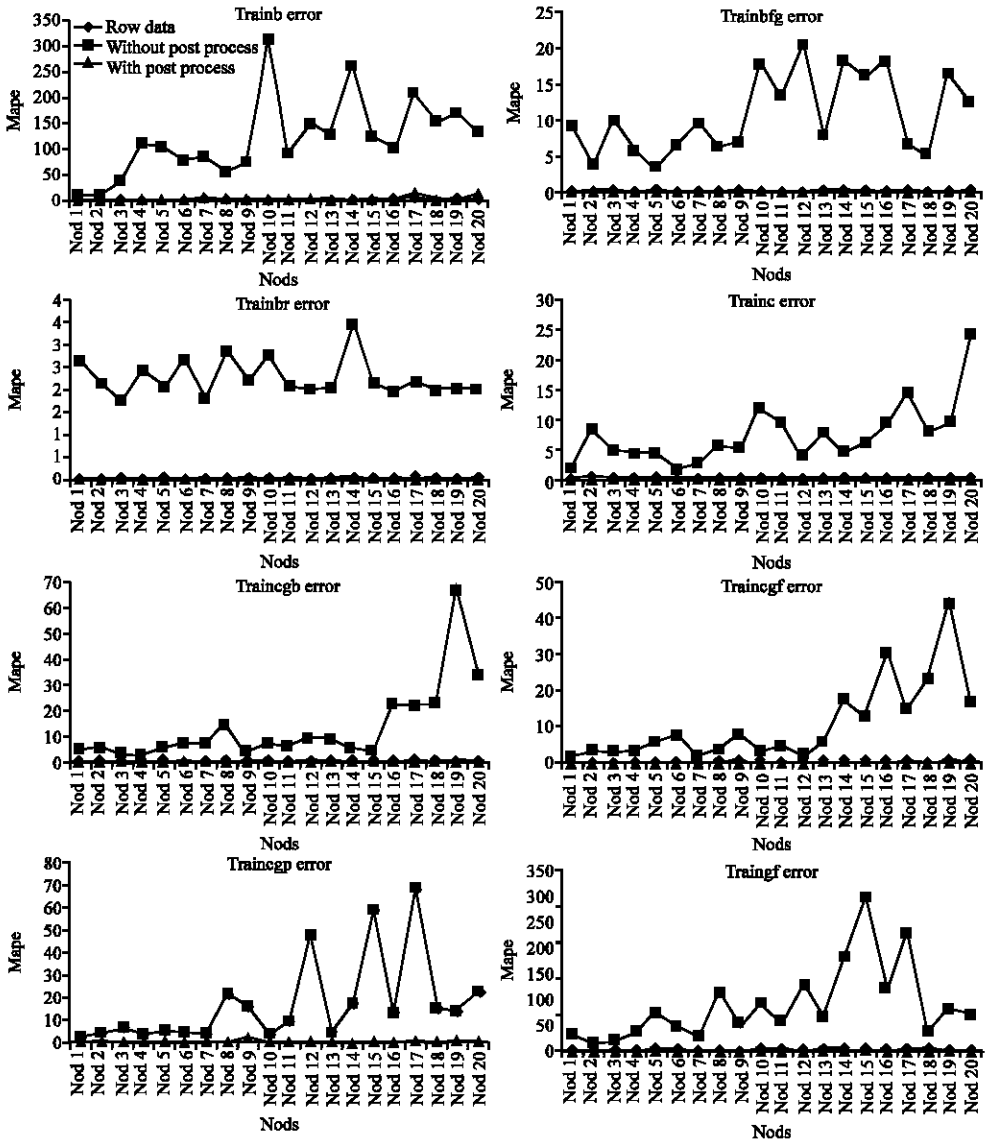

Fig. 4: Continued

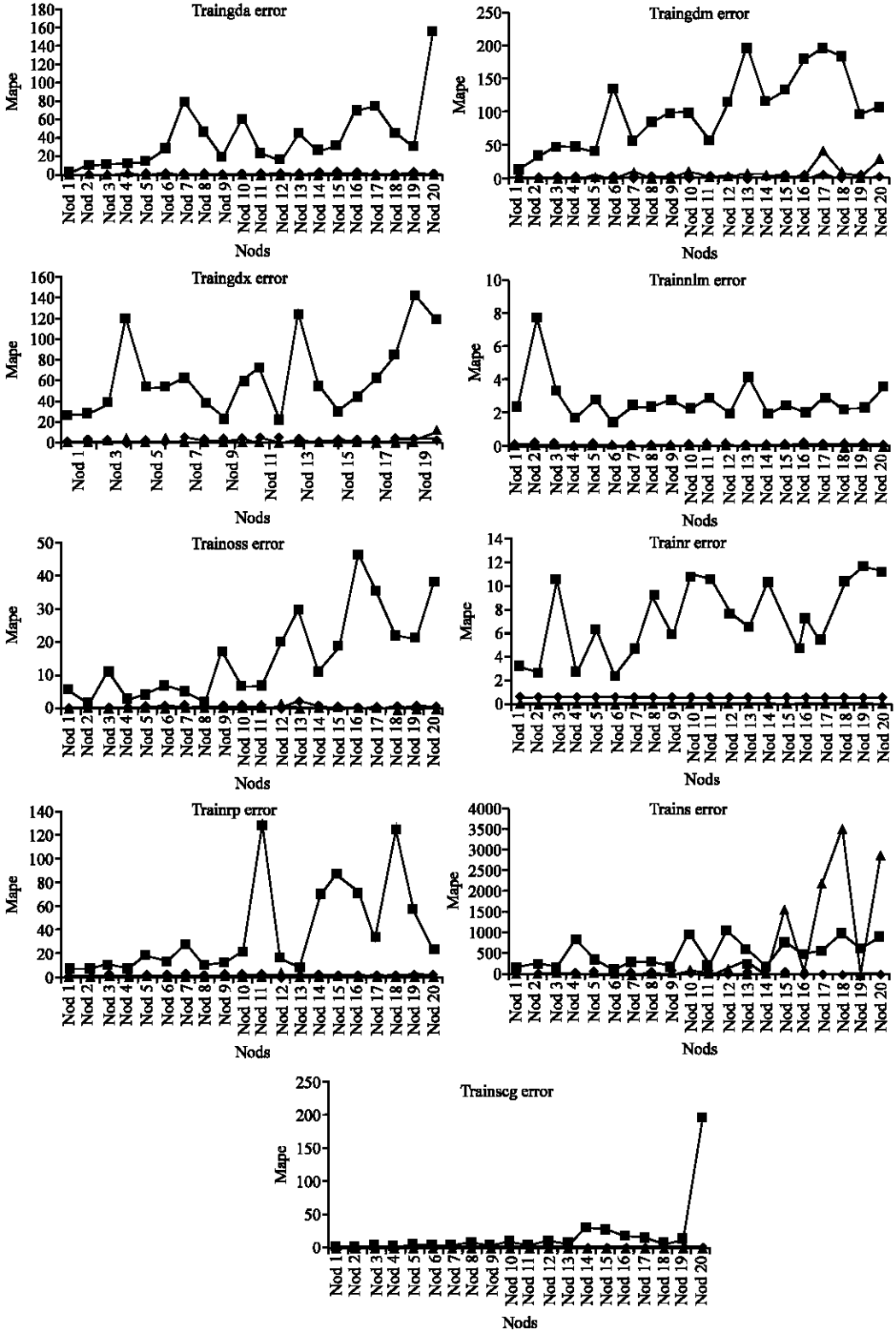

Fig. 4: MAPE values for each learning algorithm row data, with post process and without post process result

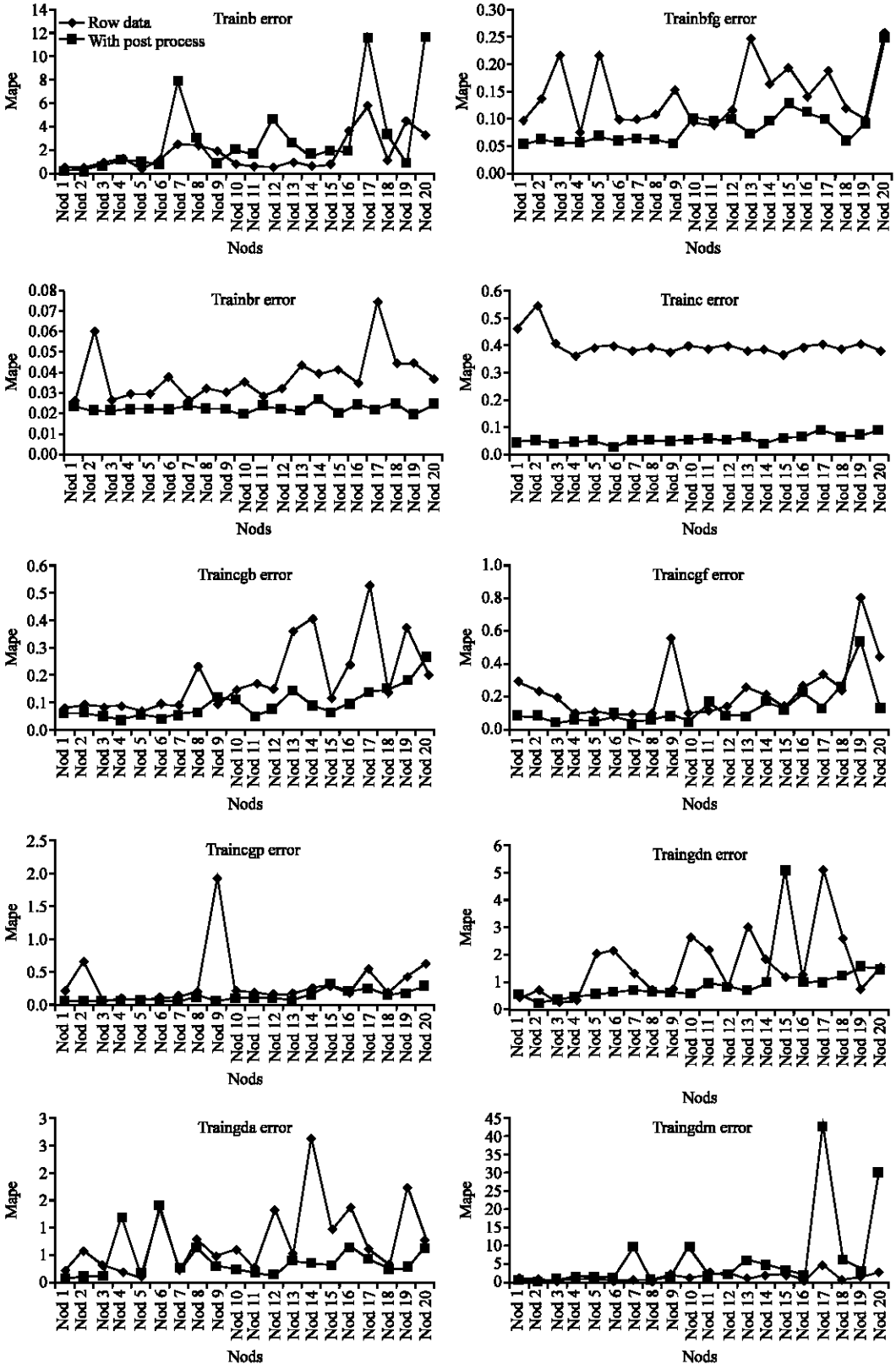

Fig. 5: Continued

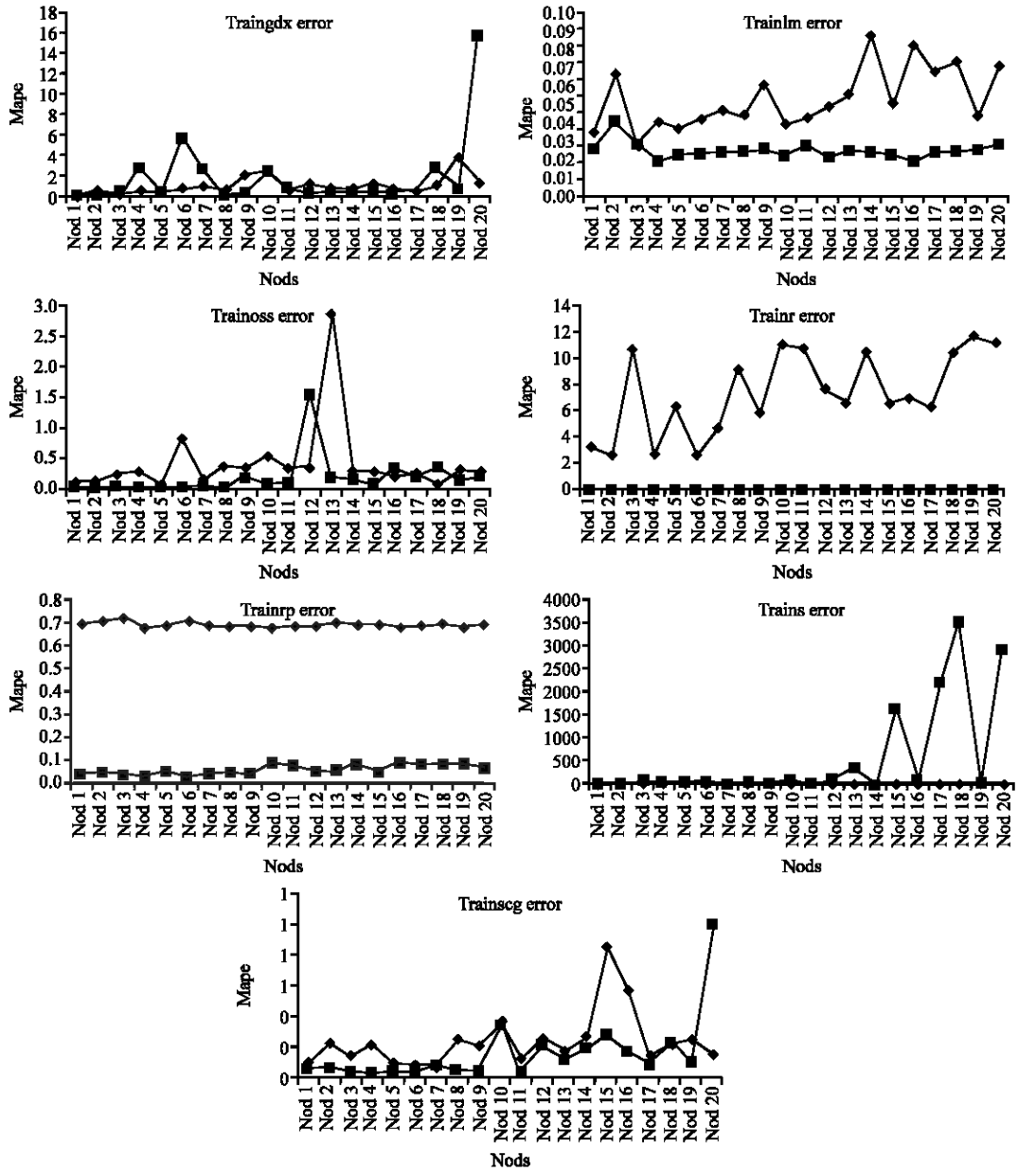

Fig. 5: MAPE values for each learning algorithm before post processing and after post processing result

Kruskal-Wallis Test on C1

Post-main	N	Median	Ave Rank	Z
1	17	0.03900	13.5	-2.32
2	17	0.08100	21.5	2.32
Overall	34		17.5	

H = 5.41 DF = 1 p = 0.020
H = 5.41 DF = 1 p = 0.020 (adjusted for ties)

Fig. 6: Kruskal-Wallis test versus row data-with post process

Kruskal-Wallis Test on C3

Pre-post	N	Median	Ave Rank	Z
1	17	0.03900	9.0	-4.98
3	17	2.55800	26.0	4.98
Overall	34		17.5	

H = 24.77 DF = 1 p = 0.000
H = 24.78 DF = 1 p = 0.000 (adjusted for ties)

Fig. 7: Kruskal-Wallis test versus without post process-with post process

Table 4: Trainlm train algorithm data with one node

Data test	1	2	3	4	5	6	7	8	9	10	11	12
ANN data	-0.015	0.005	0.027	0.031	0.031	0.013	-0.014	-0.042	-0.035	-0.002	0.009	-0.005
Actual data	-0.011	0.005	0.057	0.056	0.044	0.017	-0.023	-0.074	-0.038	0.006	0.003	0.000
ANN MAPE error (%)	42	6	52	46	30	23	40	44	10	131	218	990

Table 5: Min MAPE value of 20 nodes implemented in 17 train functions

	Node									
	1	2	3	4	5	6	7	8	9	10
Min	0.023	0.021	0.021	0.019	0.022	0.022	0.023	0.022	0.022	0.019
Max	4.168	5.751	32.94	42.98	18.12	24.65	9.602	34.46	1.182	117.38
Ave	0.351	0.417	2.086	2.954	1.306	2.038	1.440	2.346	0.304	7.841
Std	0.992	1.375	7.954	10.34	4.349	5.987	2.878	8.303	0.378	28.317
	Node									
	11	12	13	14	15	16	17	18	19	20
Min	0.023	0.021	0.021	0.024	0.020	0.019	0.022	0.024	0.019	0.024
Max	2.406	125.75	354.43	8.358	1622.2	226.87	2189.4	3479.6	38.61	2912.3
Ave	0.540	7.995	21.537	1.059	98.08	22.368	133.23	205.60	2.798	207.64
Std	0.758	30.37	85.799	2.191	392.85	63.684	529.98	843.70	9.261	703.75

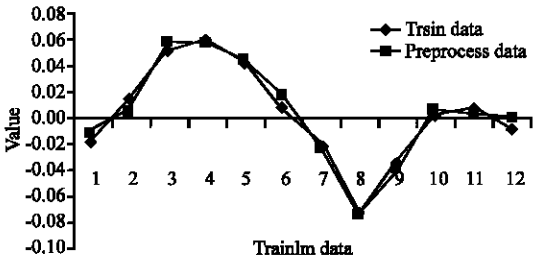

Fig. 8: Comparison of ANNW output with actual data

do not post process that value of MAPE is 279%. This value shows the necessity of ANN post process after using preprocessed data. Truly, if ANN results do not post process, the original ANN performance is ignored. The value of MAPE after ANN result post process is 2.5%.

Table 5 is yield from value in column of Table 1. Truly, in Table 5, we focused on performance of each node. With regard to Table 5 and Min variable, nodes 4, 10, 16 and 19 are the best among other nodes. As nodes 4 Max variable is less than others, is the best. With regard to our expected too many nodes suffer from over fitting problem.

CONCLUSION

In this study, all type of ANN-MLP models is examined to predicting airline number of passenger. A non-covariance stationary process was converted to covariance stationary process by a suitable data preprocessing method. Then, ACF method was used for input selection. We compared all models of ANN modeling by using DEA method. The result of DEA showed that ANNs with Levenberg-Marquardt back-propagation and Bayesian regularization back-propagation algorithms and preprocessed data are the best. So, with regard to our data, preprocessed data has positive impact on ANN performance. The result of Kruskal-Wallis Test approves yield result. Also the result of Kruskal-Wallis Test shows post process of ANN output is vital.

In order to extend the proposed model, the impact of data preprocessing and post processing in another method (such as fuzzy regression, neuro-fuzzy, particle swarm, ant colony, genetic algorithm) can also be studied.

REFERENCES

Azadeh, A., S.F. Ghaderi and S. Sohrabkhani, 2007a. Forecasting electrical consumption by integration of neural network, time series and ANOVA. Applied Math. Comput., 186: 1753-1761.

Azadeh, A., S.F. Ghaderi, S. Tarverdian and M. Saberi, 2007b. Integration of artificial neural networks and genetic algorithm to predict electrical energy consumption. Applied Math. Comput., 186: 1731-1741.

Aznarte, J.L., J.M.B. Sanchez, D.N. Lugilde, C.D.L. Fernandez, C.D. Guardia and F.A. Sanchez, 2007. Forecasting airborne pollen concentration time series with neural and neuro-fuzzy models. Expert Syst. Appl., 32: 1218-1225.

Box, G.E.P. and G.M. Jenkins, 1976. Time Series Analysis: Forecasting and Control. 1st Edn., Holden Day, San Francisco, ISBN: 0-8162-1104-3 .

Charnes, A., W.W. Cooper and E. Rhodes, 1978. Measuring the efficiency of decision making units. Eur. J. Operat. Res., 2: 429-444.

Chiang, W.C., T.L. Urban and G.W. Baldridge, 1996. A neural network approach to mutual fund net asset value forecasting. Omega, 24: 205-215.

Cybenko, G., 1989. Approximation by super-positions of a sigmoidal function. Math. Control Signal., 2: 303-314.

Enders, W., 2004. Applied Econometric Time Series. 1st Edn., John Wiley and Sons, New York, ISBN: 0471230650.

Gareta, R., L.M. Romeo and A. Gil, 2006. Forecasting of electricity prices with neural networks. Energ. Convers. Manage., 47: 1770-1778.

Hill, T., M. O'Connor and W. Remus, 1996. Neural network models for time series forecasts. Manage. Sci., 42: 1082-1092.

Hwarng, H.B., 2001. Insights into neural-network forecasting of time series corresponding to ARMA (p; q) structures. Omega-Int. J. Manage. Sci., 29: 273-289.

Indro, D.C., C.X. Jiang, B.E. Patuwo and G.P. Zhang, 1999. Predicting mutual fund performance using artificial neural networks. Omega-Int. J. Manage. S., 27: 373-380.

Jain, A. and A.M. Kumar, 2007. Hybrid neural network models for hydrologic time series forecasting. Applied Soft Comput., 7: 585-592.

Karunasinghe, D.S.K. and S.Y. Liong, 2006. Chaotic time series prediction with a global model artificial neural network. J. Hydrol., 323: 92-105.

Kim, T.Y., K.J. Oh, C. Kim and J.D. Do, 2004. Artificial neural networks for non-stationary time series. Neurocomputing, 61: 439-447.

Kohzadi, N., M.S. Boyd, B. Kermanshahi and I. Kaastra, 1996. A comparison of artificial neural network and time series models for forecasting commodity prices. Neurocomputing, 10: 169-181.

Nayak, P.C., K.P. Sudheer, D.M. Rangan and K.S. Ramasastri, 2004. A neuro-fuzzy computing technique for modeling hydrological time series. J. Hydrol., 291: 52-66.

Niska, H., T. Hiltunen, A. Karppinen, J. Ruuskanen and M. Kolehmainena, 2004. Evolving the neural network model for forecasting air pollution time series. Eng. Applied Artif. Intel., 17: 159-167.

Palmer, A., J.J. Montano and A. Sese, 2006. Designing an artificial neural network for forecasting tourism time series. Tourism Manage., 27: 781-790.

Powell, M.J.D., 1977. Restart procedures for the conjugate gradient method. Math. Program., 12: 241-254.

Stern, H.S., 1996. Neural networks in applied statistics. Technometrics, 38: 205-214.

Tseng, F.M., H.C. Yu and G.H. Tzeng, 2002. Combining neural network model with seasonal time series ARIMA model. Technol. Forecast. Soc. Change, 69: 71-87.

Zhang, G. and M.Y. Hu, 1998. Neural network forecasting of the British pound/US dollar exchange rate. Omega-Int. J. Manage. S., 26: 495-506.

Zhang, G.P., 2001. An investigation of neural networks for linear time-series forecasting. Comput. Operat. Res., 28: 1183-1202.

Zhang, G.P. and M. Qi, 2005. Neural network forecasting for seasonal and trend time series. Eur. J. Operat. Res., 160: 501-514.

Zhu, J., 1998. Data envelopment analysis versus. Principal component analysis: An illustrative study of economic performance of Chinese cities. Eur. J. Operat. Res., 111: 50-61.