Iranian Journal of Mathematical Sciences and Informatics
Vol. 4, No. 2 (2009), pp. 33-48
DOI: 10.7508/ijmsi.2009.02.004

Bank efficiency evaluation using a neural network-DEA method

G. Aslani[a], S. H. Momeni-Masuleh[*,a], A. Malek[b] and F. Ghorbani[b]

[a]Department of Mathematics, Shahed University, P.O.Box: 18151-159, Tehran, Iran
[a]Department of Mathematics, Tarbiat Modares University, P.O.Box: 14115-175, Tehran, Iran

E-mail: momeni@shahed.ac.ir

Abstract. In the present time, evaluating the performance of banks is one of the important subjects for societies and the bank managers who want to expand the scope of their operation. One of the non-parametric approaches for evaluating efficiency is data envelopment analysis(DEA). By a mathematical programming model, DEA provides an estimation of efficiency surfaces. A major problem faced by DEA is that the frontier calculated by DEA may be slightly distorted if the data is affected by statistical noises. In recent years, using the neural networks is a powerful non-parametric approach for modeling the nonlinear relations in a wide variety of decision making applications. The radial basis function neural networks (RBFNN) have proved significantly beneficial in the evaluation and assessment of complex systems. Clustering is a method by which a large set of data is grouped into clusters of smaller sets of similar data. In this paper, we proposed RBFNN with the K-means clustering method for the efficiency evaluation of a large set of branches for an Iranian bank. This approach leads to an appropriate classification of branches. The results are compared with the conventional DEA results. It is shown that, using the hybrid learning method, the weights of the neural network are convergent.

Keywords: Data envelopment analysis; Neural networks; Efficiency; Multilayered perceptron; Radial basis function; K-means clustering method.

2000 Mathematics subject classification: 90C05, 92B20, 91C20.

*Corresponding Author

Received 20 April 2009; Accepted 16 September 2009

1. Introduction

The improvement of performance for the banks in public or private sectors is an important way for any country's progress. Measuring and evaluating the operating efficiency of bank branches requires analytic techniques that provide insights beyond those available from accounting ratio analysis. Banks have aggressively sought to improve their performance by improving cash management and marketing new services that attract additional funds. Therefore, evaluating the efficiency requires employing different approaches for determining the efficiency frontier; and there are many researches in this field. Two competing methods for constructing frontiers are parametric and non-parametric methods. The chief property of non-parametric methods is that no explicit functional form needs to be imposed on data.

DEA is one of the best non-parametric methods, which creates virtual units to act as benchmarks for measuring comparative efficiency. On the other hand, artificial neural networks (ANNs) are also known as powerful systems with wide applications. ANNs have their basis in the study of complex behavior of human brain. McCulloch and Pitts [8] presented a first simple neuron. Rosenblatt [11] introduced the multi-layered structure which is known as perceptron. RBFNNs were introduced into neural network literature by Broomhead and Lowe [4]. These networks are an alternative to multi-layered perceptron because of their special structure.

In some cases, the existence of noise in data causes deviation in the DEA frontier. ANNs have been viewed as a good simulation tool to approximate numerous non-parametric and nonlinear problems. RBFNNs are widely used in neural network applications such as signal processing and pattern recognition [3, 10]. They are able to model complex mappings, while perceptron neural networks can only model by means of multiple intermediary layers. The combination of neural networks and DEA was first proposed by Athanassopoulos and Curram [1]. They used DEA as a preprocessing tool to filter the data for the training of the neural networks. Wu et al. [13] used the combination of neural networks and DEA to examine the relative efficiency of branches of a big Canadian bank. They followed Athanassopoulos and Curram's approach to preprocess the training data; and by selecting the efficient subset for learning, they guaranteed the monotonicity assumption [9]. The K-means clustering algorithm is used for the first training phase of RBF [12]. This leads to an appropriate classification of branches. Because of the good sampling nature of the data selected by this method, the network generalization capability is enhanced and the training time is considerably reduced.

This paper combines RBFNN with the K-means clustering method for the evaluating the performance of a large set of branches for an Iranian bank, which is not yet established. To the best of our knowledge, combination of RBFNN,

K-means clustering and DEA method cannot be found in prior papers.
The rest of this paper is organized as follows: Section 2 addresses the important factors in evaluating the performance of bank branches. Section 3 and Section 4 are devoted to a brief review of DEA and neural networks, respectively. Results and discussion are presented in Section 5 and conclusions are given in Section 6.

2. Bank branch performance evaluation

The relative efficiency of 280 bank branches of a big Iranian bank are analyzed here in four cities.
For studying the efficiency of an organization, its objectives and resources are required to be identified. The outputs would include those services that management believes are basic to the purpose of the organization, e.g. those services that a branch manager is expected to provide to customers. The inputs should reflect the resources that are required to produce the outputs, such that an increase (decrease) in the amount of inputs used is expected to result in an increase (decrease) in output levels. Branches are assumed to utilize 6 inputs and 8 outputs for a period of one year. We aggregate them into six categories, the list of which is given in Table 1 and Table 2.

Total expenses: = Personnel+ Office expenses + Supply costs,
Supplies: =Space + Computer terminals,
General expenses: =Rent,

Deposits: = Commercial investment deposits + Retail investment + Commercial deposits +Retail deposits+ Commercial accounts,
Loans: = Commercial loans + Retail loans,
General services: = Revenues.

Normalized input and output values for some selected branches are given in Table 3. A statistical summary of the basic characteristics of these units is provided in Table 4. We used the data for each branch for a one-year period.

3. Data envelopment analysis

Data envelopment analysis is a non-parametric linear programming technique for evaluating efficiencies of individual decision making units (DMUs). DEA identifies the best-practice frontier as the envelope of the observed production possibilities. It can handle multiple input and multiple output models. It also does not need the assumption of functional form for the relationship between the inputs and outputs of DMUs.
Suppose that for a group of n DMUs, $X = (x_{1j}, x_{2j}, \ldots, x_{mj})$ and $Y = (y_{1j}, y_{2j}, \ldots, y_{sj})$ are vectors of inputs and outputs of the jth DMU which can be represented by DMU_j. The DMU under evaluation is represented by

DMU_0. Charnes et al. [5] introduced the DEA model as a linear program that can be demonstrated as:

$$\textit{CCR} \quad \textit{Model}$$

$$\begin{aligned} &\mathrm{M}in \ \ \theta_0 - \varepsilon(\sum_{i=1}^{m} s_i^- + \sum_{r=1}^{s} s_r^+) \\ &\mathrm{s}.t. \\ &\quad \sum_{j=1}^{n} \lambda_j x_{ij} + s_i^- = \theta x_{i0}, \qquad i = 1, \ldots, m, \\ &\quad \sum_{j=1}^{n} \lambda_j y_{rj} - s_r^+ = y_{r0}, \qquad r = 1, \ldots, s, \\ &\quad \lambda_j \geq 0, \ \ s_i^- \geq 0, \ \ s_r^+ \geq 0, \qquad j = 1, \ldots, n, \\ &\quad \varepsilon \geq 0 \quad \textit{and} \quad \theta_0 \ \textit{free}. \end{aligned} \tag{1}$$

where θ_0 is the technical efficiency score for DMU_0, x_{ij} and y_{rj} are the ith input and rth output of DMU_j, respectively, λ_j are decision variables, s_i^- and s_r^+ are slack variables, and $\varepsilon \geq 0$ is a small non-Archimedean quantity.
This model has the constant returns to scale (CRS) assumption. In 1984, Banker et al. [2] presented the BCC-model with the variable returns to scale (VRS) assumption.

$$\textit{BCC} \quad \textit{Model}$$

$$\begin{aligned} &\mathrm{M}in \ \ \theta_0 - \varepsilon(\sum_{i=1}^{m} s_i^- + \sum_{r=1}^{s} s_r^+) \\ &\mathrm{s}.t. \\ &\quad \sum_{j=1}^{n} \lambda_j x_{ij} + s_i^- = \theta x_{i0}, \qquad i = 1, \ldots, m, \\ &\quad \sum_{j=1}^{n} \lambda_j y_{rj} - s_r^+ = y_{r0}, \qquad r = 1, \ldots, s, \\ &\quad \sum_{j=1}^{n} \lambda_j = 1, \\ &\quad \lambda_j \geq 0, \ \ s_i^- \geq 0, \ \ s_r^+ \geq 0, \qquad j = 1, \ldots, n, \\ &\quad \varepsilon \geq 0 \quad \textit{and} \quad \theta_0 \ \textit{free}. \end{aligned} \tag{2}$$

DMU_0 is efficient if the following conditions are satisfied:
(i) θ_0^*=1,
(ii) the slack variables $s_i^{-^*}$ and $s_r^{+^*}$ are equal to zero, for all i and r.
For each inefficient DMU, DEA can introduce a reference set containing efficient DMUs as an improvement pattern. In Section 5, we apply DEA to obtain an initial evaluation of the performance of the branches a big Iranian bank.

4. Artificial neural networks

Artificial Neural Networks (ANNs) are well known as a powerful and flexible computational tool. Their structure is inspired by the human brain and nervous system. ANNs are widely used because of their capacities. In particular, the nonlinear nature of neural networks makes them suitable for performing a variety of tasks such as prediction, function approximation, pattern classification, and forecasting. The preference of using ANNs comes from the capacity of

exploration from the observed data without any assumption regarding underlying relationships. In a neural network, multiple neurons are interconnected with synaptic weights in parallel layers. The primary significance of a network is "learning" or " training". The process of training involves incrementally changing the connection weights until the network learns to produce the correct output. The final weights are the optimal parameters of the network. One major type of neural network models that have been successfully used for various tasks is the feed-forward network type. The commonest type of artificial neural network consists of three kinds of layers: a layer of "input" units is connected to a layer of "hidden" units, which is in turn connected to a layer of "output" units. In this paper, we use perceptron and radial basis function, which belong to the multi-layered feed-forward neural networks group.

4.1. **Multi-layered perceptron networks.** The multi-layered perceptrons (MLPs) have been applied to solve many complex problems by being trained in a supervised manner with a highly popular algorithm known as error back-propagation [11, 10]. The Levenberg-Marquardt (LM) training method [6] is one of the second-order methods for error minimization that provides faster training for medium or small networks. However, for very large networks, the memory requirement of this algorithm is high. Fig. 1 shows the structure of an MLP, where each circle represents an individual neuron and the connective weights and biases are denoted by W_{ij} and b_i, respectively.

4.2. **Radial basis function neural networks.** The radial basis function neural networks (RBFNNs), were presented into the neural networks literature by Broomhead and Lowe in 1988 [3, 4]. The basic architecture of RBFNN consists of three fully connected layers. The RBF neurons in the hidden layer use the radial basis function as the transfer function, and the inputs to the last, output layer is a linear combination of the outcomes of the hidden neurons. The most used transfer function for the hidden neurons of RBFNN is the Guassian function. Let $S = \left\{ (x^\mu, y^\mu) \middle| \quad x^\mu \in \Re^r, \quad y^\mu \in \Re, \quad \mu = 1, \ldots, n \right\}$ be the set of n observations. The output of each RBF unit is:

$$\phi_j(x^\mu) = exp\ (-\frac{\|x^\mu - c_j\|}{\sigma_j^2}), \quad j = 1, \ldots, r, \quad \mu = 1, \ldots, n, \tag{3}$$

where ϕ_j denotes the output of the jth RBF neuron, c_j and σ_j are the center and width of the jth RBF neuron, $x^\mu \in R^r$ is the μth input vector of pattern learning and $\|.\|$ indicates the Euclidean norm. The response of the network with one neuron in the output layer is computed as follows:

$$y(x^\mu) = \sum_{j=1}^{r} w_j \phi_j(x^\mu) + b, \tag{4}$$

where y is the output of the network for input vector x^μ, w_j is the weight connection between the output unit and the jth hidden layer unit, and b is the bias for the output unit.

Although the training of multi-layered perceptrons is usually performed by the back-propagation algorithm, the RBF network can be trained in many different ways. The hybrid learning method for RBF can be divided into three separate stages [12] by selecting centers, widths, and weights. In this paper, we combine unsupervised and supervised strategies for the training process of the network. In the first step, for unsupervised learning, we choose centers and widths for hidden neurons. One efficient unsupervised algorithm is the clustering algorithm. It categorizes the input patterns into a finite number of groups. A very simple and known clustering method is the K-means algorithm [7]. The K-means algorithm starts with K centers (initial values are selected randomly). Each pattern in the data set is assigned to the closest center, and finally the centers are recalculated according to the associated patterns. This process is repeated until a stopping criterion is satisfied. The algorithm has the following steps:

(i) Initialize the K centers randomly.

(ii) For each pattern Z_p, in the data set, compute its membership $u(m_k|Z_p)$ in each cluster with a center m_k and its weight $W(Z_p)$.

(iii) Recalculate the K centers by the following general formula:

$$m_k = \frac{\sum_{\forall Z_p} u(m_k|Z_p)\, W(Z_p)\, Z_p}{\sum_{\forall Z_p} u(m_k|Z_p)\, W(Z_p)} \tag{5}$$

until stopping criterion is satisfied. Continue the algorithm until no noticeable changes are observed in the centers m_k.

Note that $d^2(Z_p, m_n) = min\{d^2(Z_p, m_j) \mid j = 1, \ldots, K\}$ in which d represents the Euclidean distance. The membership for K-means is defined as:

$$u(m_k|Z_p) = \begin{cases} 1, & k = n, \\ 0, & otherwise, \end{cases}$$

and the weight function for all patterns $W(Z_p) = 1$.

Assigning the widths of the hidden neurons is the next stage. They are usually determined by the P-nearest neighbor algorithm. The widths for each hidden neuron is computed as the average Euclidean distance from the center to the P nearest centers. Denote the center of neuron j by m_j, then its width is computed as follows:

$$\sigma_j = \frac{1}{P} \sum_{i=1}^{P} |m_j - m_i|, \qquad j = 1, \ldots, K, \tag{6}$$

where m_i are the P-nearest centers to m_j, and K is the number of the hidden neuron. The value of P is chosen by users.

In the second step of the training algorithm, we set the initial weights of the

output layer at random values. The error is obtained by comparing the output of the network against the desired output. The weights of the output layer are adjusted by the supervised strategy by:

$$\triangle w_{kj} = \eta \sum_{\mu=1}^{M} \phi_j(x^\mu)(y_k^\mu - F_k^\mu), \tag{7}$$

where F_k^μ and y_k^μ are the actual and target output, respectively, $\phi_j(x^\mu)$ is the output of the jth hidden neuron, η is the learning rate, and $x^\mu \in R^d$ is the μth input vector [12]. The numerical results for training process of the RBFNN are demonstrated in section 5. Fig. 2 depicts an RBF network with two neurons in the input and one neuron in the output layer.

The two-phase algorithm for training RBF can be summarized as follows:

(1) Find the centers of the hidden layer by the K-means clustering algorithm. Compute the widths by the P-nearest neighbor method.
(2) Adjust the output layer weights by formula (7).

5. Results and discussion

For training neural networks, 80% of input patterns are used. To assess simulation model accuracy, the mean squared error (MSE) is used. Hereafter, we refer to these sets as learning sets. The stages of the neural network-DEA method are as follows:

Stage 1): The efficiency scores are computed by the CCR (input-oriented) model. To obtain the preprocessed data set, they are divided by an integer 1000 for the purpose of uncomplicated training and simulation.

Stage 2): By applying the trained MLP, we determine the efficiency scores. The computed efficiency results using DEA-MLP are demonstrated in Fig. 3.

Stage 3): In order to analyze the performance of the RBF network, we use four data sets and consider 80% of them as the learning set. S_1 is the data set for one city and the other data sets are construct by adding the information of other cities to S_1. The characteristics of data sets are given in detail in Table 5. The structures of the networks under study are given in Table 6. The efficiency scores are determined by the trained RBF. Fig. 4 presents the results of the efficiency computed by DEA-RBF.

Fig. 5 demonstrates the changes of MSE in four data sets S_1, S_2, S_3, and S_4. It shows that by increasing the number of branches in each data set, MSE is reduced. Figs. (6-9) illustrate the convergence of the output weights in the RBFNN for different number of DMUs. The convergence is shown by exhibiting the trajectories network of the weights. These Figures show that by starting from a random value for each weight, the network will converge in less than 50

iterations. By a comparison of the number of epochs between MLP and RBF, it can be seen that convergence happens in a smaller number of iterations in RBF. Regression analysis results of DEA, DEA-RBF, and DEA-MLP, and their comparison are shown in Figs. (10-11).

6. Conclusions

We have investigated the performance of radial basis function network and the multi-layered perceptron network for evaluating the branch efficiency of a big Iranian bank. The comparisons on parameter settings between the MLP and RBF neural network models are listed in Table 6. In both processes, DEA is used to help in preprocessing the data for training the networks. The RBF-DEA technique proposed in this paper shows an efficient way of evaluating the branch efficiency in terms of speed training, memory storage, and CPU time taken to complete the simulation, since it uses a smaller number of neurons and epochs. Here, with the proposed RBF-DEA model capable of handling a large bank sample, we hope to develop a general powerful numerical tool for evaluating the bank branch efficiency.

Table 1. Details of input data.

Input Data	
Total expenses	*Personnel, Office expenses and Supply costs*
Supplies	*Space and Computer terminals*
General expenses	*Rent*

Table 2. Details of output data.

Output Data	
Deposits	*Commercial deposits, Retail deposits, Commercial investment deposits, Retail investment and Commercial accounts*
Loans	*Commercial loans and Retail loans*
General services	*Revenues*

TABLE 3. Normalized the input and output values for sample branches.

Branch	Expenses	Supplies	General expenses	Deposits	Loans	General services
1	0.131835	0.309886	0.073586	0.127699	0.17509	0.157728
10	0.177761	0.38375	0.263157	0.144063	0.355913	0.184215
20	0.2968	0.575681	0.14035	0.127166	0.334921	0.241903
30	0.211948	0.386363	0.284113	0.096625	0.087979	0.1312557
40	0.354293	0.309659	0.08382	0.232506	0.594234	0.221732
50	0.130199	0.298863	0.245614	0.064789	0.287397	0.116472
60	0.217072	0.52272	0.631578	0.083914	0.119463	0.137977
100	0.331245	0.22145	0.22457	0.33145	0.203564	0.21457
120	0.02457	0.03321	0.200145	0.40023	0.220124	0.3364
160	0.33654	0.22547	0.40025	0.12458	0.030857	0.45
200	0.55412	0.3354	0.57841	0.332145	0.10021	0.22145
220	0.15969	0.15924	0.16366	0.91434	0.77501	0.42099
240	0.59769	0.65656	0.38291	0.04306	0.58148	0.76384
260	0.7	0.23758	0.87417	0.86073	0.6516	0.87414
280	0.51467	0.84369	0.63633	0.95156	0.26027	0.51467

TABLE 4. Statistical properties for the normalized inputs and the outputs of branches.

Data	Min	Max	Average	Standard deviation
Expenses	0.0012	0.7768	0.2719	0.1379
Supplies	0.0124	0.9654	0.3409	0.1781
General expenses	0.0110	1.0000	0.2895	0.1839
Deposits	0.0112	0.9901	0.2974	0.2110
Loans	0.0012	1.0000	0.3268	0.2165
General services	0.0124	1.0000	0.3467	0.2401

TABLE 5. Detailed analysis for different data sets: S1, S2, S3 and S4.

Data set	Number of branches	Size of learning set	Learning rate	Mean square error
S_1	100	80	0.5	0.118603
S_2	200	160	0.1	0.06172
S_3	250	200	0.1	0.054917
S_4	280	224	0.1	0.052778

TABLE 6. Structure of networks.

Network architecture	Multi-layer perceptron	Radial basis function
Number of neurons: input-hidden-output	6-10-1	6-6-1
Activation function: hidden/output	tanh /linear	radial basis/linear
Learning algorithm	Levenberg-Marquardt	hybrid
Epoch (Max)	1000	200

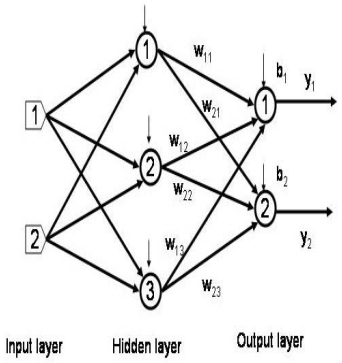

FIGURE 1. MLP neural network structure with three neurons in hidden and two neurons in output layer.

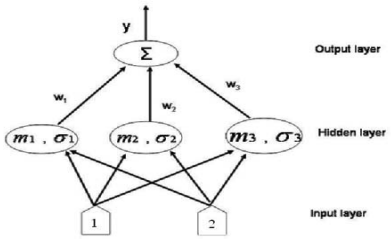

FIGURE 2. RBF neural network structure with three neurons in hidden and one neuron in output layer.

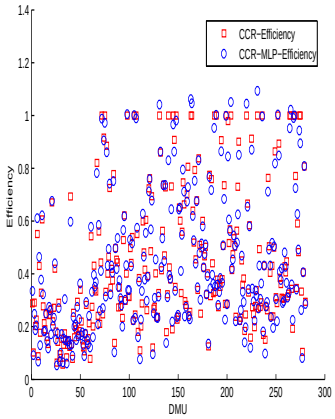

FIGURE 3. The comparison between result of CCR and CCR-MLP: the squares show the efficiency score computed by CCR and the circles represent the efficiency score computed by CCR-MLP.

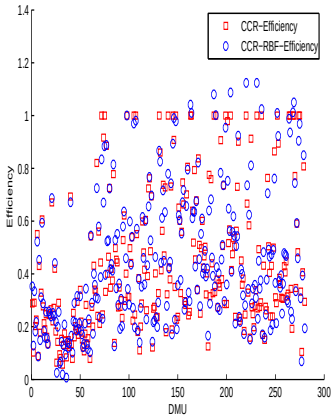

FIGURE 4. The comparison between result of CCR and CCR-RBF: the squares denote the efficiency score computed by CCR and the circles represent the efficiency score computed by CCR-RBF.

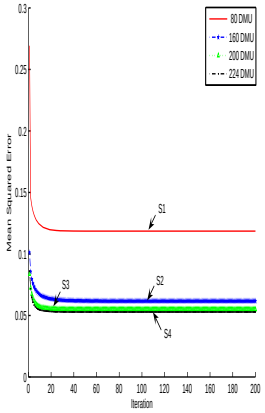

FIGURE 5. Trajectories of changes in the mean squared error in RBFNN for data sets S_1, S_2, S_3, and S_4.

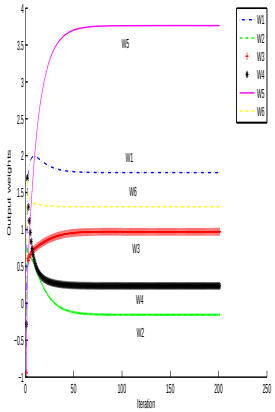

FIGURE 6. The convergence of output weights for S1 for 80 DMUs in the training stage.

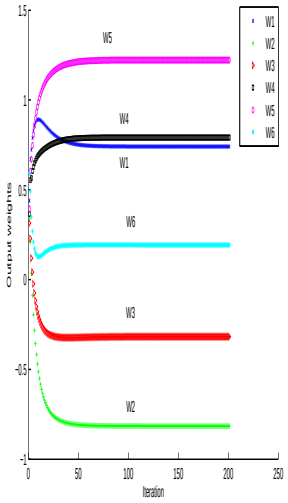

FIGURE 7. The convergence of output weights for S2 for 160 DMUs in the training stage.

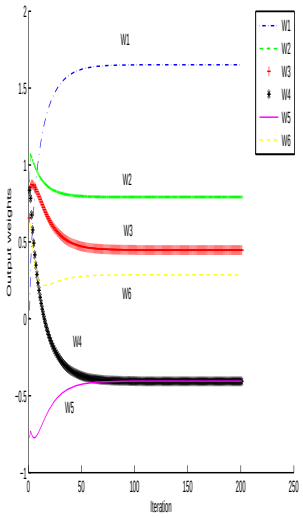

FIGURE 8. The convergence of output weights for S3 for 200 DMUs in the training stage.

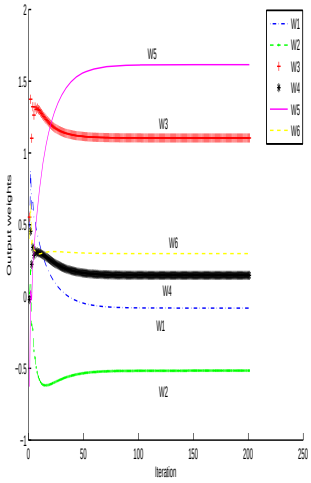

FIGURE 9. The convergence of output weights for S4 for 224 DMUs in the training stage.

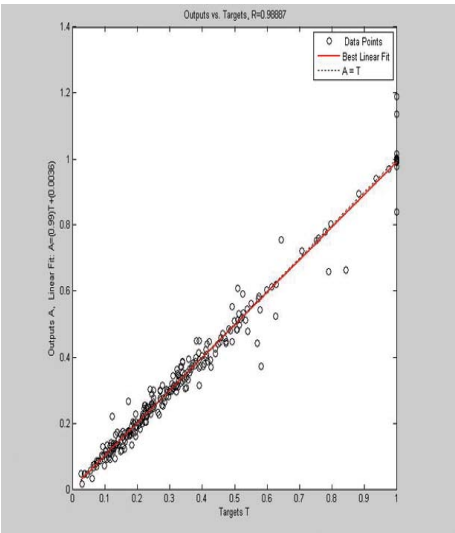

FIGURE 10. Regression analysis results and their comparison between DEA and DEA-RBF.

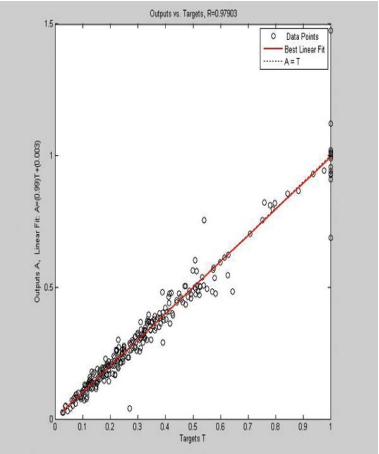

FIGURE 11. Regression analysis results and their comparison between DEA and DEA-MLP.

REFERENCES

[1] A. Athanassopoulos and S. P. Curram, A comparison of data envelopment analysis and artificial neural networks as tool for assessing the efficiency of decision making units, *J. Operational Research Society*, **47** (8) (1996), 1000-1016.

[2] R. D. Banker, A. Charnes, and W. W. Cooper, Some models for estimating technical and scale inefficiencies in data envelopment analysis, *J. Management Science*, **30** (1984), 1078-1092.

[3] A. G. Bors, Introduction to radial basis function networks, *J. IEEE Transaction on Neural Networks*, **2** (2) (2001), 302-309.

[4] D. S. Broomhead and D. Lowe, Multivariable functional interpolation and adaptive networks, *Complex Systems*, **2** (1988), 321-355.

[5] A. Charnes, W. W. Cooper, and E. Rhodes, Measuring the efficiency of decision making units, *J. European Journal of Operational Research*, **6** 2 (1978), 429-444.

[6] M. T. Hagan and M. B. Menhaj, Training feedforward networks with the marquardt algorithm, *J. IEEE Transaction on Neural Networks*, **5** (6) (1994), 989-993.

[7] J. A. Hartigan and M. A. Wang, A K-means clustring algorithm, *J. Applied Statistics*, **28** (1979), 100-108.

[8] W. McCulloch and W. Pits, A logical calculus of the ideas immanent in nervous activity, *Bulletine of Mathematical Biophysics*, **5** (1943), 115-133.

[9] P. C. Pendharkar and J. A. Rodger, Technical efficiency-based selection of learning cases to improve forecasting accuracy of neural networks under monotonicity assumption, *Decision Support Systems*, **36** (1) (2003), 117-136.

[10] Ph. Picton, *Neural networks*, Palgrave, second edition, 2000.

[11] F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain, *Pshchological Review*, **65** (1958), 386-408.

[12] F. Schwenker, H. A. Kestler, and G. Palm, Three learning phases for radial basis function networks, *J. Neural Networks*, **14** (2001), 439-458.

[13] D. Wu, Z. Yang, and L. Liang, Using DEA-neural network approach to evaluate branch efficiency of large canadian bank, *J. Expert Systems with Applications*, **31** (2006), 108-115.