Iranian Economic Review, Vol.11, No.16, Spring 2006

Estimation of Industrial Production Costs, Using Regression Analysis, Neural Networks or Hybrid Neural - Regression Method?

Esmaiel Abounoori*

Morteza Bagherpour**

Abstract

Estimation (Forecasting) of industrial production costs is one of the most important factor affecting decisions in the highly competitive markets. Thus, accuracy of the estimation is highly desirable. Hibrid Regression Neural Network is an approach proposed in this paper to obtain better fitness in comparison with Regression Analysis and the Neural Network methods. Comparing the estimated results from Regression Analysis and Neural Networks with the Hybrid Neural-Regression method has indicated the superiority of the latter method.

Keywords: Regression Analysis, Neural Networks, Hybrids Neural–Regression Method, Comparison, Estimation

I- Introduction

Cost estimation is performed regularly throughout the life cycle of many products. In the early phases of the life cycle, when a new product is considered, cost estimation analyses are used to support the production decision. When

---

*- Corresponding Author, Associate Professor of Economics and Social Statistics, Department of Economics, University of Mazandaran, Babolsar-Iran. Email:abounoories@yahoo.com

**- Department of Socio-economics System Engineering, Mazandaran University of Science & Technology, Babol-Iran. Email:morteza13592002@yahoo.com

alternative configurations are considered, the best alternative will be selected based on its estimated life cycle cost and benefits. In the production phase the estimated cost of manufacturing is an important factor affecting all decisions. Also, the decision to replace a system of production is based on the estimated cost of its future operation and maintenance. In general, estimation of future costs are required at each phase to support the decision making process. The cost of estimation, the accuracy of the estimates and the time it takes to produce cost estimates are important factors influencing the ability of organizations that develop, manufacture, sell and support products to survive in today's highly competitive markets.

In the published paper in International Journal of Production Economics, Shtub(1999) described a cost estimating system that can be linked to the CAD data. The proposed system was based on a Neural Network learning how to modify cost estimates when a new technology is developed. He compared the neural network with linear regression analysis used for cost estimation.

In this paper, a generalized regression neural network model was selected to forecast cost estimation. In the proposed network, linear regression & neural network solutions are used as inputs and actual cost is considered as desired output of the network. A comparative analysis showed that the proposed method (hybrid neural–regression) is more efficient and the output of network is closely equal to desired out put (target).

2- A Brief Literature Surveys

The increasing intensity of cost based competition motivated a significant number of researches aimed at developing tools that support the cost estimation analysis. Number of most cost estimation is based on information about the product, the inputs and the production processes required for manufacturing. Cochran (1976) suggested the use of regression analysis for cost estimation. In his approach a multi variable linear regression model is used, in which the dependent variable is cost and the independent variables are the factors of production. Dean (1989) proposed a similar cost estimation model. Unlike the regression analysis that minimizes the root mean squares (RMS) of errors, Deans model minimizes the Euclidean distance between the estimated cost and

its actual value. Son (1991) proposed a cost estimation model in which the total cost is the sum of relatively well structured cost (RWSC) and relatively ill structured cost (RISC). The first category includes the cost of direct labor and materials, the cost of machines and equipment, the cost of facilities etc. The second category includes the costs of quality and costs related to setup, idle time etc. Some cost estimating models are designed to estimate the cost of specific operations in the manufacturing process. A cost estimation methodology is developed by Boothroyed and Dewhurst (1990) in order to evaluate different design alternatives for assembled products. He recommends the best design is the one that minimizes the cost of assembly. Shtub and Zimmerman (1993) suggested a modification of the DFA method. They suggested a cost estimation model that supports the selection of the best assembly system for a product or a family of products. Using the Boothroyed and Dewhurst (1991) approach as the source of data, Shtub and Zimmerman (1993) developed a neural network model that estimates the cost of assembly by different assembly systems.

The main purpose of this paper is to compare regression analysis (RA), artificial neural networks (ANN) with Generalized regression neural networks (GRNN).

2-1- Regression Analysis

Method includes the experimental investigations, mathematical methods and statistical analysis, as well as econometrics. A researcher either intends to investigate relation between independent variables or parameters to determine the coefficients of independent variables to obtain optimal solution for determining objective function. The regression coefficients determined in the beginning are estimated by using the experimental data.

2-2- Artificial Neural Networks

According to Filho, Cabral and Soarces (1998), ANNs presented by McCulloch and Pitts (1943), in which ANNs are born from approach of developing intelligent systems by simulating the biological structure and the work of the human brain. Afterwards, the number of studies on ANNs increased considerably. As indicated in Grossmand and Thursby (1995) the theory of ANN is based on neurobiology, mathematics and physics. An ANN is composed of hierarchically organized neural bundles, bound parallel to each other. Unlike

the classical systems, the use of ANN is based on their previous experiences. Since information is processed in a parallel fashion by the neurons in ANNs, the system is much faster than the classical systems. Human body consists of trillions of cells. A portion of them is the nerve cells called ''neurons''. These neurons have different shapes and sizes. The neurons are the cell; made specially for conducting information in an electrochemical way. A biological neuron is composed of a body, which contains the nucleus and two types of extensions called ''dendrite'' and ''axon''. The nucleus is located at the center of the neuron and provides energy for cellular activities. A neuron is connected to other neurons via axons and dendrites. The canals which bring impulses to the nerve cells are called as the dendrite; the canals conducting impulses to other cells are called as the axon. Dendrites receive the impulses by contacting with other neurons and conduct these impulses to the nucleus. The impulse output from nucleus is conducted via axon and this operation is repeated continuously. The touch surfaces between two neurons are called ''synapse''. The impulses conducted by the axon of neuron are conducted to other neuron by synapses.

2-2-1- Radial Basis Function Network (RBFN)

A Radial Basis Function Network (RBFN) is a hybrid learning neural network, which is a two-layer fully-connected network with an input layer which performs no computation[1].

The hidden layer: Learning in the hidden layer is performed by using an unsupervised method, the K-means algorithm[2]. First, the user must choose a number of centers and this number will correspond to the number of neurons in the hidden layer. The K-means algorithm is used to position the centers in the best way, i.e. so that each presented record is attached to its nearest center (or cluster). As it is an unsupervised learning method, only the inputs data are presented to the K-means algorithm. The radial basis functions are then applied to each center: There is one radial Gaussian function for each hidden unit which simulates the effect of overlapping and locally tuned receptive fields. The activation function of the hidden nodes is radially symmetric in input space; the

---

1- Moody and Darken (1989).
2- See the K-Means technical overview.

magnitude of the activation given a particular record is a decreasing function of the distance between the input vector of the record and the center of the basis function. The distance metric employed is the Euclidean distance. The activation of the hidden neuron *i* with presented pattern s have the form:

$$\phi_S(i) = \exp\left(\frac{\|s - c_i\|}{2\sigma_i^2}\right) \qquad (1)$$

Each hidden unit has an associated σ 'width' value which defines the nature and scope of the unit's receptive field response. It is equivalent to the standard deviation of the width of the Gaussian response, so larger values allow more points to be included. Figure 1 shows the shape of this activation function in one input dimension, taking $c_i = 0$ and $\sigma_i = 1$:

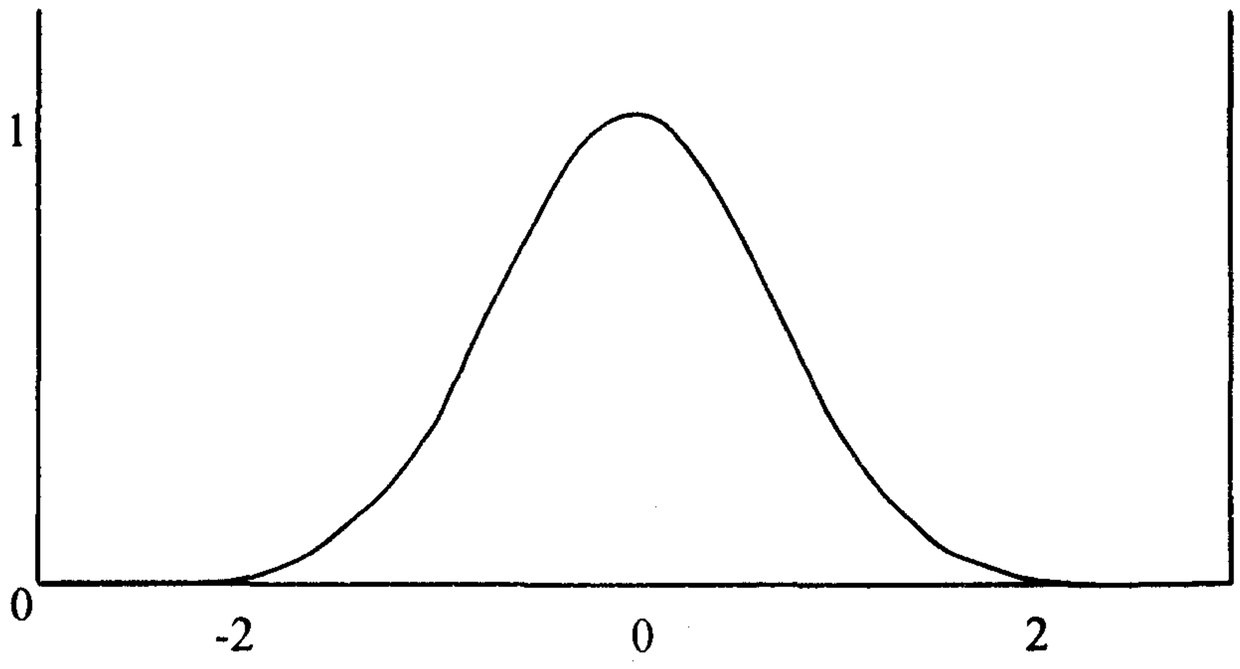

Figure 1: A Shape of Activation Function

The closer the presenting patterns to the center, the higher the response. This response will always be between 0 and 1 (The response will be 1 only if the presented pattern is identical to the center). The width of the 'bell' shape depends on the value of σ: The bigger the value of σ, the larger the width of the 'bell' shape. The value of σ is computed according to the distance between the corresponding cluster and its 2 closest one. The user can not neither choose the value of σ nor the way it's computed.

The role of the hidden units is to perform a non-linear transformation of the input space into the space of activations of the hidden units. It is this representation that gives the RBF a much greater representational power than the linear perception.

The output layer: Learning in the output layer in performed by computing a linear combination of the activation of the basis functions, parameterised by the weights **W** between hidden and output layers. For the output *k* of record *s*, where *j* describes the j^{th} hidden neuron, i.e. the j^{th} center, the number of centers being *m*, we have the predicted output:

$$o_k(s)=\sum_{j=1} W_{jk}\times\varphi_s(j) \tag{2}$$

The weights W_{jk} are initialized to small random values (between -0.001 and 0.001). After that, they are fine-tuned at each iteration *t* using the *supervised* learning method Least Mean Square (LMS):

$$W_{jk}(t)=W_{jk}(t-1)+\Delta W_{jk}(t) \tag{3}$$

Where:

$$\Delta W_{jk}(t)=\eta\times(r_k-o_k)\times\varphi_s+\alpha\Delta W_{jk}(t-1) \tag{4}$$

$W_{jk}(t)$ is the weight from hidden unit j to output unit k at the iteration t, and $\Delta W_{jk}(t)$ is the weight adjustment.

η is a trial-independent learning rate and α (also called the momentum).

η can vary according to t but α is generally a constant between 0 and 1.

$r_k - o_k$ is in fact the prediction error of the j^{th} output unit.

These operations are repeated until the maximum number of iterations fixed is reached or until the prediction error $r_k - o_k$ is less than a given threshold. Figure 2 referents the RBFN shape:

The RBFN Parameters: It's the number of clusters computed by the K-Means algorithm. This parameter is very important and must be chosen carefully according to the size of the data used to train the network. For instance, if the data contain about 500 records, then the number of centers can be between 20 and 500 (though 500 will over-train the network), but if the number of records is close to 500000, it's better to take at least 500 centers. The default value is 20, because it is big enough to permit the network to cope with many different records, but small enough to avoid overtraining. The K-Means algorithm used into RBFN is the same used for building the K-Means node. The difference is

that for RBFN, there are no options available to the user to change the parameters of the K-Means algorithm. The number of iterations is fixed to 10,

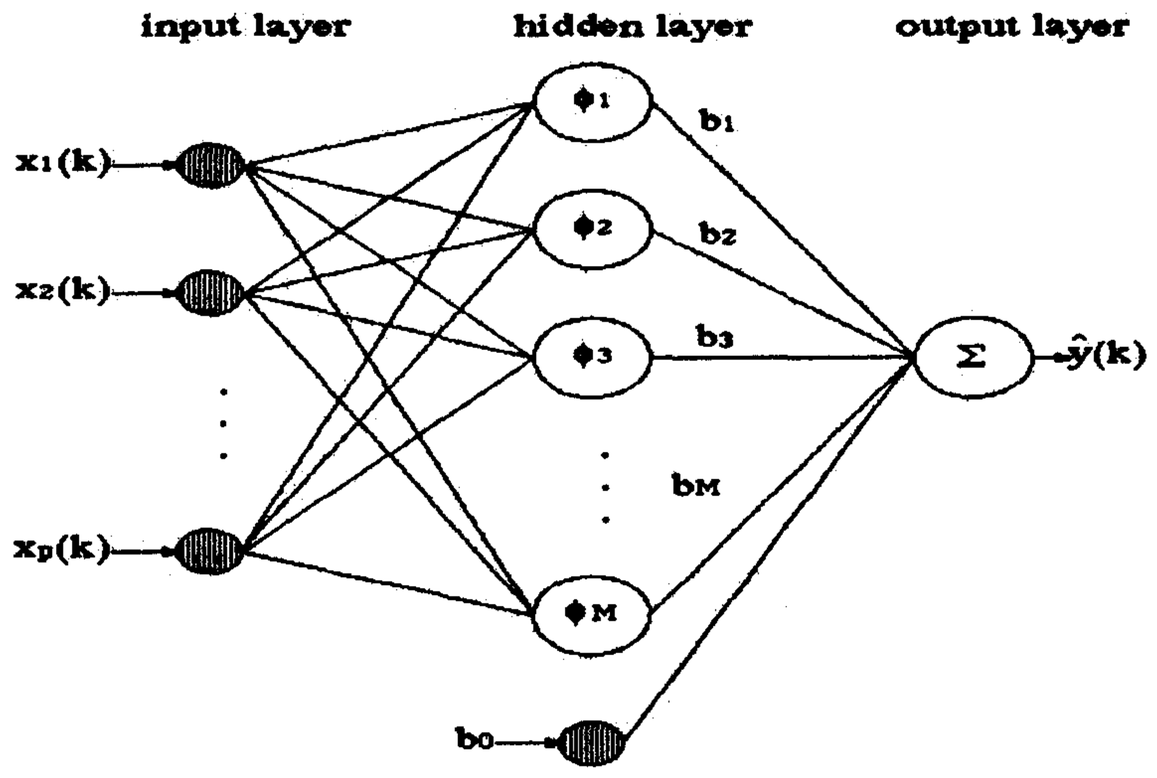

Figure 2: Typical Radial Basis Function (RBF) Network

and the change tolerance to 0.000001. Symbolic data are not encoded in the way they are in the K-Means node. The encoding follows the Back-prop convention.

Learning Rates: Unlike Back-prop, the learning rates in RBFN are both fixed.

Also, it's very important to choose a good value, especially for eta. If the value is too big, the NN won't learn due to exponentially growing weights, and if it's too small the NN will learn very slowly. The good value for eta is the value for which the network learns fast without having its weights growing to an infinite value.

The user can either put a value for eta, or this value can be computed automatically.

The eta value won't usually change the accuracy of a network, but will have an effect on the time the network needs to learn[1].

2-2-2- Generalized Regression Neural Network (GRNN)

General Regression Neural Network is a normalized RBF network in which there is a hidden unit centered at every training case. These RBF units are called

---

1- Specht (1991).

"kernels" and are usually probability density functions such as the Gaussian. The hidden-to-output weights are just the target values, so the output is simply a weighted average of the target values of training cases close to the given input case. The only weights that need to be learned are the widths of the RBF units. These widths (often a single width is used) are called "smoothing parameters" or "bandwidths" and are usually chosen by cross-validation or by more esoteric methods that are not well-known in the neural net literature; gradient descent is not used.

GRNN is a universal approximate for smooth functions, so it should be able to solve any smooth function-approximation problem given enough data. The main drawback of RNN is that, like kernel methods in general, it suffers badly from the curse of dimensionality. GRNN cannot ignore irrelevant inputs without major modifications to the basic algorithm. So GRNN is not likely to be the top choice if you have more than 5 or 6 non-redundant inputs.

In general, GRNN is a special extension of radial basis function network (RBFN). It is a feed forward neural network based on nonlinear regression theory consisting four layers: the input, the pattern (hidden) layer, the summation layer, and the output layer. It can approximate any arbitrary mapping between input and output vectors. Topology of the generalized regression neural network is shown in figure 3.

While the neurons in the first three layers are fully connected, each product neuron is connected only to some processing units in the summation layer. The function of the input and pattern layers of the GRNN is exactly the same as it is in the RBFN. The summation layer has two different types of processing units: the summation unit and a single division unit. The number of the summation units is always the same as the number of GRNN output units; their function is

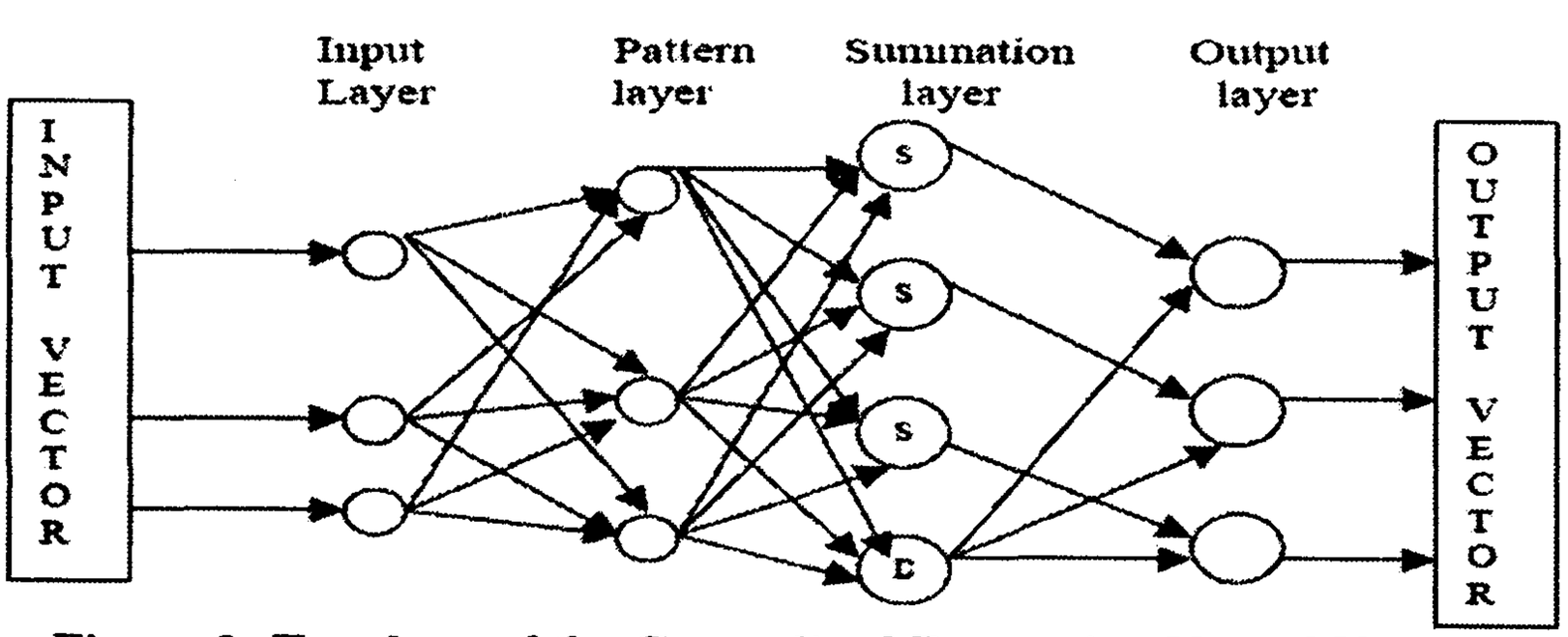

Figure 3: Topology of the Generalized Regression Neural Network

essentially the same as the function of the output units in the RBFN. The division unit only sums the weighted activation of the pattern units without using any activation function. Each of the GRNN output units is connected only to its corresponding summation unit and to the division unit; there are no weights in these connections. The function of the output units consists in a simple division of the signal coming from the summation unit, followed by the signal coming from the division unit. The summation and output layers together basically perform a normalization of the output vector, thus making the GRNN much less sensitive to the proper choice of the number of the pattern units that the RBFN. The overlapping of radial basis function of individual pattern units is not a problem for the GRNN; in fact, it turns out to be an important parameter allowing the user to influence generalization capabilities of the GRNN. In general, larger values of the width of radial basis function of the pattern units results in a smoother interpolation of the output vectors values among the values corresponding to the center of the redial basis functions of the individual pattern units. The training of the GRNN is quite different from the training used for the RBFN. It is completed after presentation of each input – output vector pair from the training set to the GRNN input layer only once, that is, both the centers of the redial basis functions of the pattern unit and the weights in connections of the pattern units and the processing units in the summation layer are assigned simultaneously. The training of the pattern units is unsupervised, as in the case of the RBFN, but employees a special clustering algorithm which makes it unnecessary to define the number of pattern unit in advance.

Simultaneously with building the pattern layer, the values of the weights in connections between the neurons in the pattern layer and the summation layer are also set using the supervised training. The weights in connection between each pattern unit and the individual summation units are directly assigned with values identical to the elements of the output vector corresponding in the training set to the input vector which formed the center of the redial basis function of that particular pattern unit.

In case that some additional input vectors in the training set are assigned to the same pattern unit, values of the elements of their corresponding output vectors are simply added to the pervious values of these weights. At the same

time the weight in the connection of each pattern unit and the division unit, which was originally set to zero, is increased by one for each input vector from the training set which is assigned to this pattern unit.

3- Problem Methodology

Since the cost estimation is a regular job during the life cycle of any firms in industry, adaptation & evaluation of the approach within the production system, seems to be natural. Thus the use of artificial neural networks seems to be reasonable. In this section a Generalized Regression Neural Network (GRNN) is selected among the other neural network models. GRNN usually uses to function estimation, and also uses from normal distribution as a transfer function, and it is good advantage to cover real world problems more. In designed GRNN, the output of neural network (NN) & regression analysis (RA) -that are available in Shtub - are considered as input vectors and actual cost is defined as target of GRNN. The proposed hybrid approach is shown in figure 4.

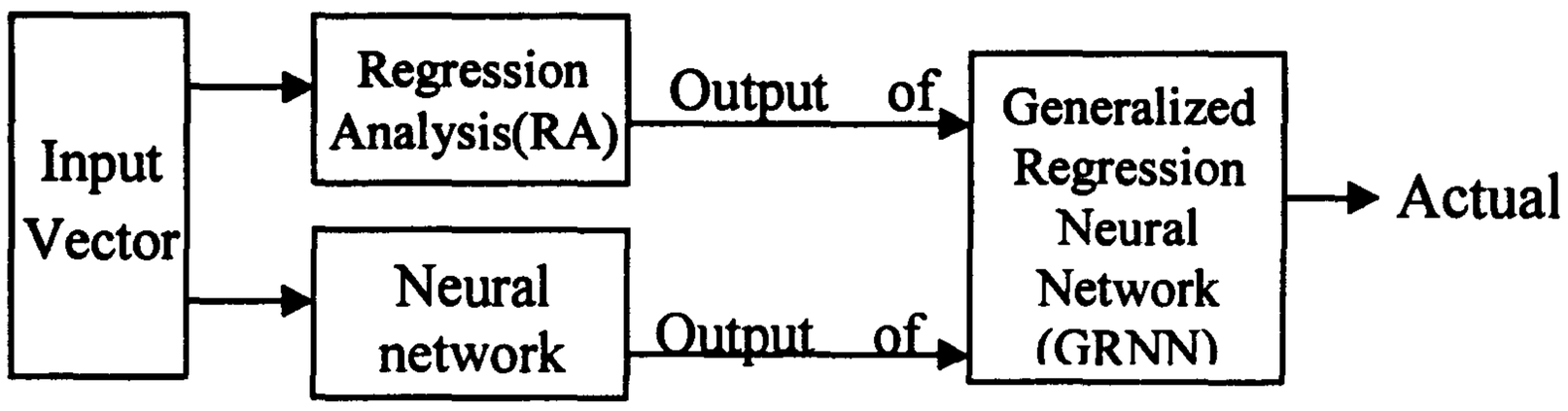

Figure 4: The Proposed Hybrid Neural-Regression Approach

4- Comparative Analysis

The results of Cost estimation is evaluated based on three methods, that of the regression analysis and neural network results are used from Shtub (1999, p. 206, Table 5) and the result of Hybrid Neural–Regression method is discussed in this paper. Table 1 illustrates comparative analysis among these three methods. Squared Error (SE) as well as the Sum of Squared Error (SSE) are presented for each estimation method.

Table 1: Comparative Analysis Among three Different Methods

Obs.	Actual Cost	Regression Analysis		Artificial Neural Networks		Hybrid Neural-Regression	
		Est.	SE	Est.	SE	Est.	SE
1	239.5	149.3	8136.04	157	6806.25	239.500	0
2	106.2	99.3	47.61	124	316.84	106.200	0
3	408	312.4	9139.36	419.4	129.96	408.000	0
4	156.6	163.8	51.84	150.8	33.64	156.600	0
5	433.5	301	17556.25	365.4	4637.61	433.500	0
6	109	151.2	1780.84	109.3	0.09	109.000	0
7	157.2	151	38.44	140.9	265.69	157.200	0
8	101.6	137.1	1260.25	117.8	262.44	101.600	0
9	170.4	139.3	967.21	154	268.96	170.400	0
10	210	311.1	10221.21	187.1	524.41	210.000	0
11	148.6	163.5	222.01	163.3	216.09	148.600	0
12	289.4	231.7	3329.29	304.5	228.01	289.400	0
13	166	101.4	4173.16	97.5	4692.25	166.000	0
14	306	273.3	1069.29	275	961	306.000	0
15	127.2	132.3	26.01	156.5	858.49	127.200	0
16	153.8	172.6	353.44	159.7	34.81	153.722	0.00606841
17	425	326.2	9761.44	358.7	4395.69	425.000	0
18	118.5	143.7	635.04	96	506.25	118.500	0
19	162.5	232.6	4914.01	170.3	60.84	162.500	0
20	112.7	175.6	3956.41	195.9	6922.24	112.778	0.00606841
21	412	300.8	12365.44	377.1	1218.01	412.000	0
22	170.8	107.2	4044.96	153.9	285.61	170.800	0
23	200.1	223.8	561.69	175.4	610.09	200.100	0
24	88	67.7	412.09	120.9	1082.41	88.000	0
25	113.9	122.6	75.69	131.9	324	113.900	0
26	151	178.5	756.25	172.2	449.44	151.000	0
27	207.8	290.8	6889	276.2	4678.56	207.800	0
28	145.1	212	4475.61	163.1	324	145.100	0
29	204	193	121	144.4	3552.16	204.000	0
30	182.4	186.3	15.21	163.5	357.21	182.400	0
31	221.3	283.9	3918.76	241.8	420.25	221.300	0
32	199.3	240.3	1681	175.6	561.69	199.300	0
33	237.8	131.2	11363.56	183.1	2992.09	237.800	0
34	170.6	248.2	6021.76	159.2	129.96	170.600	0
35	84	60.7	542.89	93.9	98.01	84.000	0
36	76.1	65.1	121	71	26.01	76.100	0
ESS			131005.1		49231.06		0.01213682

Source: Estimated using MatLab software, based on the actual data obtained from Shtub (1999, p. 206).

As it is well indicated in the table 1, the Sum of Squared Error (SSE) corresponding the GRNN is about 0.01213, which is considerably smaller than that of ANN, and RA. Moreover, the coefficients of correlations between actual data and the fitted values (R) corresponding to the different methods in table 2 show the superiority of the GRNN.

Table 2: Important Decision Parameters for Each Method

	SSE	***R***	***P-Value***
Regression Analysis	131005	0.77	0.8
Artificial Neural Network	49231	0.92	0.73
Hybrid Neural - Regression	0.012	1	1

5- Conclusion

In this paper, a hybrid neural – regression method developed based on neural network and regression results available in Shtub (1999) as the network inputs. The result of proposed hybrid method is compared to the result of regression analysis and neural network available in Shtub (1999). Although statistical analysis show that no groups have means significantly different from actual data, according to minimum SSE and maximum correlation coefficient with actual data and the best P-value – the hybrid neural–regression analysis is to be chosen as the best forecasting method.

References

1- Boothroyd, G and Dewhurst, P (1990) Product Design for Assembly, Boothroyed Dewhurst, Inc., Wake"eld, RI.
2- Cochran E.B. (1976), "Using regression techniques in cost analysis Part 2", *International Journal of Production Research,* 14 (4), PP. 489-511.
3- Dean, E.B.(1989), Parametric cost estimating: a design function, 33rd. Annual Meeting of the American Association of Cost Engineers, San Diego, CA, 1989, PP. 25-28.
4- Filho, B.D.B, E.L.L. Cabral, A.J. Soares (1998), A new approach to artificial neural networks, *IEEE Trans. Neural Networks* 9 (6).
5- Grossman,B.G. and M.H. Thursby (1995), Neural network processing for fiber optic sensors and smart systems, in: Fiber Optic Smart Structures, Wiley, New York.
6- McCulloch, W. S and W. Pitts (1943), A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (1943) 115–133.
7- Mitchell, T. M (1997), Machine Learning, McGraw-Hill, Singapore.

8- Moody J., and C. Darken (1989), "Fast learning in networks of locally-tuned processing units". Neural Computation, Vol. 1, no. 2, PP. 281-294, 1989.

9- Shtub, A and Zimmerman, Y(1993)" A neural-network-based approach for estimating the cost of assembly systems", *International Journal of Production Economics* 32.

10- Shtub, Versano (1999), "Estimating the Cost of Steel Pipe Bending, a Comparison between Neural Networks and Regression Analysis", *International Journal of Production Economic*, Vol. 62, PP. 201-207.

11- Son, Y.K. (1991), A cost estimation model for advanced manufacturing systems, *International Journal of Production Research*, 29 (3), PP. 441-452.

12- Specht, D.F. (1991), "A Generalized Regression Neural Network", IEEE Transactions on Neural Networks, 2, Nov. 1991, 568-576.