Journal of Applied Sciences

ISSN 1812-5654

Journal of Applied Sciences 8 (16): 2825-2833, 2008
ISSN 1812-5654
© 2008 Asian Network for Scientific Information

A Hybrid Method of Neural Networks and Genetic Algorithm in Econometric Modeling and Analysis

Hamed Hasheminia and Seyed Taghi Akhavan Niaki
Department of Industrial Engineering, Sharif University of Technology,
P.O. Box 11155-9414, Azadi Ave., Tehran, Iran

Abstract: In this study a hybrid method of neural networks-genetic algorithms is proposed and applied in an economical case study. The results of this study show that the proposed hybrid algorithm is a more efficient modeling approach compared to either a single neural network method or a single genetic algorithm approach. Since modeling based on the observed data is also employed in other fields of science, the application of the proposed method is not restricted only to economics.

Key words: Neural networks, genetic algorithms, econometrics modeling and analysis

INTRODUCTION

Discovering the relationship among different observed phenomena is one of the most important goals of any field of science. Particularly, finding the economical relations among economical phenomena is a very important task for economists, as different econometrical and statistical approaches and concepts have been developed to ease the application of appropriate methods for estimating, modeling and statistical analysis of any experiment. In this regard, various methods such as the maximum likelihood, the method of moments, the least square method and different statistical approaches like hypothesis testing for prevailing classical econometrical and statistical approaches have been proposed in the literature (Greene, 2000; Neter *et al*., 1996).

In recent decades, the increasing power of computers and data acquisition systems have directed researchers to develop new computational and heuristic methods to solve estimation and modeling problems (Winker and Gilli 2004). Artificial neural networks (Etheridge *et al*., 2000; Lippmann, 1987), genetic algorithms (Holland, 1975; Baragona *et al*., 2001), simulating annealing (Angelis *et al*., 2001; Ahmed and Alkhamis, 2002) and threshold accepting approach (Lee *et al*., 2004; Fitzenberger and Winker, 1998) are the most prevalent of these methods. These algorithms and methods usually lead to better and more efficient solutions. However, sometimes their use is limited and their reliability is skeptical. Although, different methods, such as making confidence intervals for the response (Kilmer *et al*., 1999; Ricals and Personnaz, 2000) and some other statistical approaches based on these algorithms have been developed (Baffi *et al*., 2002), using these methods still have some limitations. For example, should anyone use a wrong set of predictive variables, which does not have any relation to the output (response variable), as an input to a neural network, the normal artificial neural network may not show any sensitivity.

In this study, based on the genetic algorithm and the neural network modeling, a hybrid method is introduced where the output vector of a genetic algorithm is the input vector of a neural network. The genetic algorithm is responsible to find the best combination of independent variables in a regression/econometric model such that the accuracy of the inputs to the neural network model is guaranteed. The neural network part of the proposed method is in charge of the learning process. Then, an economical case study is considered using the proposed methodology, whose result is compared with the cases where either the neural network or the genetic algorithm is used solely.

ARTIFICIAL NEURAL NETWORKS

One of the most effective methods for pattern classification and function mapping is the Artificial Neural Networks (ANN). An ANN is a machine that is designed to model the way the brain performs. A particular task or function of interest is usually implemented using electronic components or simulated in software on digital computers.

The performance of the biological neurons such as data/information storage is hidden in the shell of neural

Corresponding Author: Seyed Taghi Akhavan Niaki, Department of Industrial Engineering, Sharif University of Technology, P.O. Box 11155-9414, Azadi Ave., Tehran, Iran Tel: (+9821) 66165740

connections, whose function is to modify the synaptic weights of the network in an orderly fashion to reach a desired objective. The procedure used to perform the learning process is called learning algorithm.

Multilayer Preceptron Neural Networks (PNN) is perhaps the most popular network architecture in use today and is discussed at length in most neural network textbooks (e.g., Bishop, 1995). In this type of network, we arrange the units in a layered feed forward topology, where the units each perform a biased weighted sum of their inputs and pass this activation level through a transfer function to produce their output. In other words, the inputs, I_i, to each node are multiplied by the strength (Weight of the corresponding connections to the node, W_i) and then summed by a bias as:

$$\text{Net} = \sum_i I_i W_i + \text{bias} \tag{1}$$

The Net input is then sent through an activation function (or transfer function) that generates the output value for the node.

$$\text{Out} = f(\text{Net}) \tag{2}$$

The activation functions can be in the form of linear or non-linear and is chosen based on the need of the problem at hand. In this article, the sigmoid activation function is used which is formulized in Eq. 3.

$$\text{Out} = f(\text{Net}) = \frac{1}{1 + e^{-\frac{\text{Net}}{T}}}; \quad T > 0 \tag{3}$$

The amount of T (temperature rate) in the sigmoid activation function adjusts the linear part of the sigmoid function. The lesser the value of T, the more the linear part of the sigmoid activation function becomes (Bishop, 1995).

The network thus has a simple interpretation as a form of input-output model, with the weights and thresholds (biases) as the free parameters of the model. Figure 1 shows the topology of PNN with one hidden layer. Such networks can model functions of almost arbitrary complexity, with the number of layers and number of units in each, determining the function's complexity. Important issues in Multilayer Perceptrons (MLP) design include specification of the number of hidden layers and number of units in them (Bishop, 1995; Haykin, 1994).

The number of input and output units is defined by the problem and the number of hidden units to use is far from clear. A good starting point is to use one hidden layer and trade the number of units in the hidden layer.

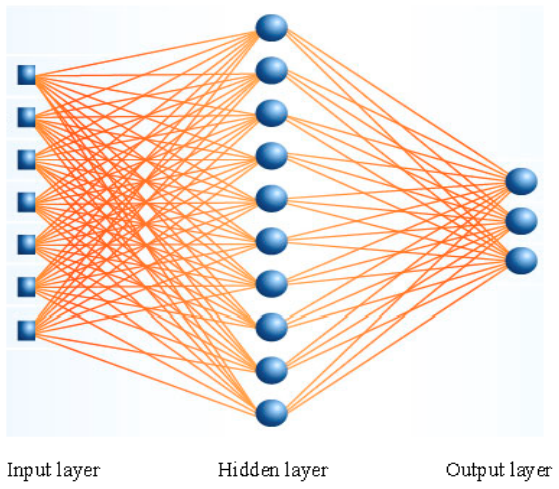

Fig. 1: The topology of PNN with a hidden layer

Once we select the number of layers and the number of units in each layer, the network's weights and thresholds must be set to minimize the prediction error made by the network.

This is the role of training algorithms. This process is equivalent to fitting the model represented by the network to the training data available. The error of a particular configuration of the network can be determined by running all the training cases through the network and comparing the actual output generated with the desired (target) outputs. Then, by an error function, we combine the differences to get the network error. The most common error function used in the literature is the Sum of Squared Error (SSE), in which we square and sum together the individual errors of output units in each case.

The best-known example of a neural network-training algorithm is the back propagation algorithm (Haykin, 1994; Patterson, 1996; Steil, 2006). In this algorithm, the learning process is done by introducing the patterns to the network forwardly and adjusting the weights and biases backwardly. Thus, these networks are referred as back propagation networks.

Although modern second-order algorithms such as conjugate gradient descent and Levenberg-Marquardt are substantially faster for many problems, back propagation still has advantages in some circumstances and is the easiest to understand (Bishop, 1995). There are also heuristic modifications of back propagation, which work well for some problem domains.

In back propagation, the gradient vector of the error surface is calculated. This vector points along the line of steepest descent from the current point, so we know that if we move along it a short distance, we will decrease the error. A sequence of such moves, slowing as we near the bottom, will eventually find a minimum of some sort. The difficult part is to decide how large the steps should be. Large steps may converge more quickly, but may also overstep the solution or go off in the wrong direction (if the error surface is very eccentric). A classic example of this in neural network training is where the algorithm progresses very slowly along a steep, narrow, valley, bouncing from one side across to the other. In contrast, although very small steps may go in the correct direction, they also require a large number of iterations. In practice, the step size is proportional to the slope (so that the algorithms settle down in a minimum) and to a special constant: the learning rate. The correct setting for the learning rate is application-dependent and is typically chosen by experiment; it may also be time varying, getting smaller as the algorithm progresses.

The algorithm therefore progresses iteratively, through a number of epochs. In each epoch, we submit the training cases in turn to the network and target actual outputs; then, compare and calculate the error. This error, together with the error surface gradient, is used to adjust the weights and then the process repeats. The initial network configuration is random and training stops when a given number of epochs elapse, when the error reaches an acceptable level, or when the error stops improving (Bishop, 1995; Haykin, 1994).

Leave-one-out processing: In order to estimate the error of a neural network, especially in cases where we have a small number of patterns, a very reliable method is the Leave-One-Out method. Since using all the patterns in the training process may lead to memorizing the patterns instead of learning them, this method is considered a very effective way of recognizing the true error of the network.

In this method, if we have n patterns, the system is repeatedly trained with (n-1) cases and then is tested with the one remaining case. This process is repeated with each case being left out of the training set once and the average or the sum of the squared errors (Leave-One-out index) is used to evaluate the performance of the network.

Although the Leave-One-Out process generates a good estimate of the true error, it is computationally expensive. Nevertheless, since the patterns in the case study of this article is very small (16 patterns), the authors applied this method to evaluate the true error of the network (Cawley *et al*., 2003; Ancona *et al*., 2006).

GENETIC ALGORITHMS

The fundamental information of living organisms is gene, which is generally a part of a chromosome determining specific characteristics of an organism. Using this fact, was lead to developing a new type of heuristic algorithms named Genetic Algorithms (GA). Since Holland (1975) introduced the principals of genetic algorithms, it has been used in several applications in different fields of science.

Typical genetic algorithms: Essentially, GA is a method of breeding computer programs and solutions to optimization or search problems by means of simulated evolution. Processes based on natural selection, crossover and mutation are repeatedly applied to a population of binary strings which represent potential solutions. Over time, the number of above-average individuals increases and better-fit individuals are created until a good solution to the problem at hand is found. A pseudo-code for a typical genetic algorithm follows:

```
1.  Generate initial population P of solutions
2.  While stopping criteria not met, do:
3.      Select $P' \subset P''$ (mating pool), initialize $P'' = 0$ (set of children)
4.      For i = 1 to n do:
5.          Select individuals $x^a$ and $x^b$ at random from P'
6.          Apply crossover to $x^a$ and $x^b$ to produce $x^{child}$
7.          Randomly mutate produced child $x^{child}$
8.          $P'' = P'' \cup Y\ x^{child}$
9.      End for
10.     P = Survive (P', P")
11. End while
```

The survivors can be formed either by the last generated individuals P', P" Y {fittest from P'}, only the fittest from P" or the fittest from P' Y P", where P" is the mating pool, P" is the set of children, x^a and x^b are individuals and x^{child} is the randomly produced child of x^a and x^b (Kim and Han, 2000; Man *et al*., 1997; Goldberg, 1989).

Genetic algorithm for econometric models: In order to find the best regression/econometric model among the candidates, we employ the algorithm proposed by Hasheminia and Niaki (2006). In this algorithm if we denote the dependent variable by Y and the independent variables by X_i; i = 1,2, ..., n, then a linear regression model may have a form in Eq. 4.

$$Y = \alpha F_1(X_1) + \beta F_2(X_2) + \ldots + \delta F_n(X_n) + \varepsilon \tag{4}$$

where, $F_1(X_1)$,..., $F_n(X_n)$ are different functional forms of the independent variables and the coefficients α, β,..., δ can be easily estimated by the least square method.

The goal of the genetic algorithm is to find models which are better in terms of the sum of squared error and by minimization of the following model, this goal is reached.

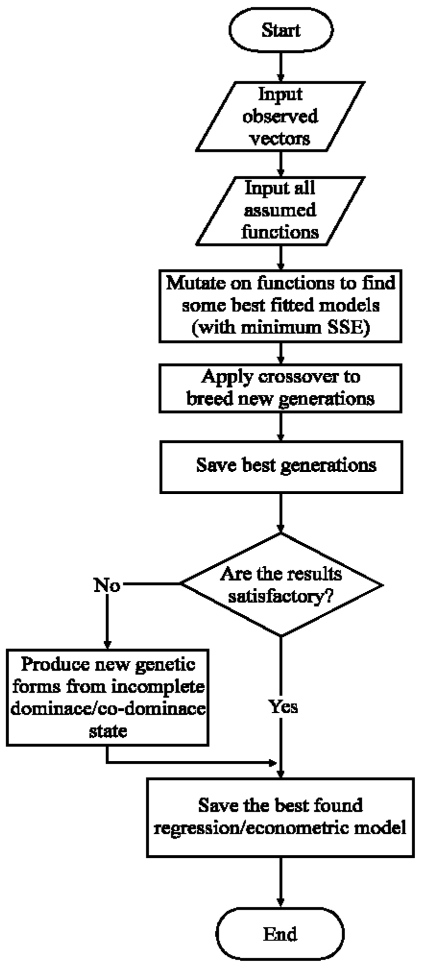
Start
Input observed vectors
Input all assumed functions
Mutate on functions to find some best fitted models (with minimum SSE)
Apply crossover to breed new generations
Save best generations
Are the results satisfactory?
No
Yes
Produce new genetic forms from incomplete dominace/co-dominace state
Save the best found regression/econometric model
End

Fig. 2: Summarized flowchart of the genetic algorithm in econometric models

$$\text{Minimize SSE} = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{o}[C_{ik}\alpha_{ik}F_{ik}(X_j) - Y_j]^2$$

Subject to:

$$\sum_{i=1}^{m}\sum_{k=1}^{o}C_{ik} = 1$$

where, X_j is the jth independent variable, Y_j is the jth observation of the dependent variable, F_{ik} is the kth function of ith independent variable, a_{ik} is the corresponding coefficient of F_{ik} and C_{ik} are dummy variables which assure us not to use more than one function of one observed vector is a single model.

The genetic algorithm of Hasheminia and Niaki (2006) finds appropriate functions (F_{ik}) and their corresponding coefficients which result in better model fitting. Figure 2 shows the flowchart of this algorithm.

THE HYBRID NEURAL NETWORK-GENETIC ALGORITHM METHOD

In the proposed hybrid algorithm, first different observation vectors and different functions are introduced to the model. By applying the genetic algorithm the best combination of functions along with their corresponding coefficients are estimated. Then, appropriate statistical tests of hypothesis are used to find whether the estimated model is statistically reliable, which means that the estimated model based on the functions of the observed vectors rather than the observation vectors themselves, is statistically meaningful. In other words, selected functions of the observed vectors are proved to affect the dependent variable which means this state of functions of observed vectors does not contain extra independent variables. After finding the functions of observation vectors, input vectors based on the selected functions from genetic algorithm are used as the input vector of the neural network.

Application of the proposed hybrid algorithm will assure the reliability of the trained network in case it is properly trained. Since by using the genetic algorithm and its corresponding statistical hypothesis testing, one can conclude that the input vector affect on the output layer, the neural network will be more reliable than when we use input vectors just based on our conjectures on the input vectors. Figure 3 shows a general framework of the proposed methodology.

A CASE STUDY

Here, we utilize the data from a case study by Hasheminia and Niaki (2006) to ease comparing the effectiveness of the genetic algorithm and the proposed hybrid algorithm.

The case study was devoted to finding the demand functions of loans in terms of the amount and the number of demands in one of the Iranian developing banks. The first step in selecting the appropriate econometric models was to find a possible set of independent variables. This set contained the Net Gross Domestic Production (NGDP) (Vera, 2002), net capital of the bank (NCAP), binary variable of advertisement (ADV), binary variable of economical crisis (EC), Unemployment Rate (UR) (Ashley, 2002), interest rate (INT) (Escandón and Díaz-Bautista, 2000), inflation (INF) (Evans, 1997) and a constant (A). In addition, two different dependent variables, the net total amount of loans (NTA) and the total number of demands for loans (TND), were studied. The corresponding data for these variables in 16 different years are shown in Table 1.

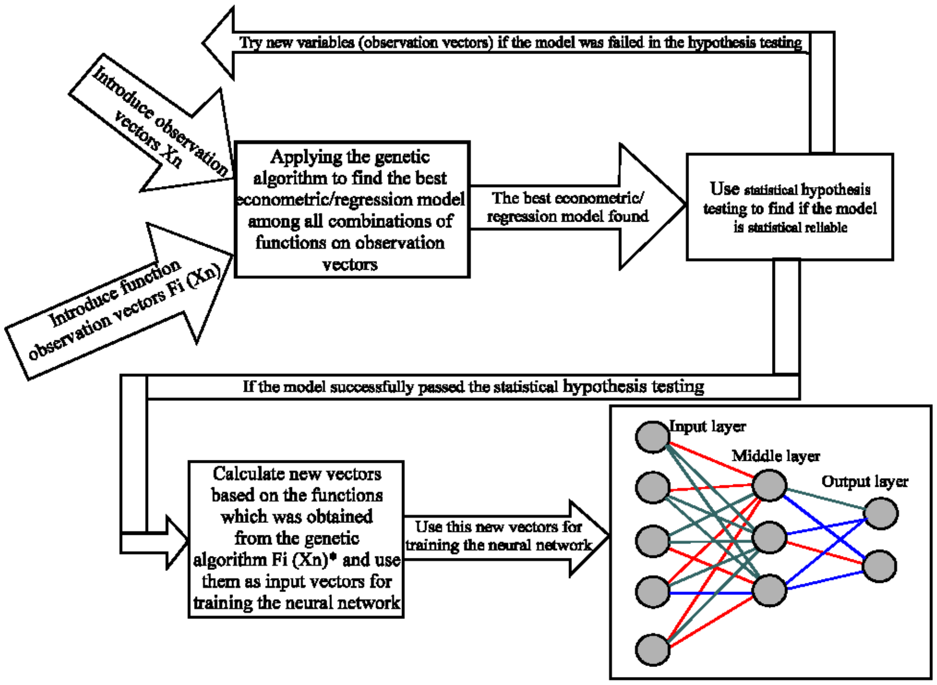
Try new variables (observation vectors) if the model was failed in the hypothesis testing
Introduce observation vectors Xn
Applying the genetic algorithm to find the best econometric/regression model among all combinations of functions on observation vectors
The best econometric/ regression model found
Use statistical hypothesis testing to find if the model is statistical reliable
Introduce function observation vectors Fi (Xn)
If the model successfully passed the statistical hypothesis testing
Calculate new vectors based on the functions which was obtained from the genetic algorithm Fi (Xn)* and use them as input vectors for training the neural network
Use this new vectors for training the neural network
Input layer
Middle layer
Output layer

Fig. 3: The framework of the proposed hybrid neural network-genetic algorithm method

Table 1: The gathered data for estimating the demand function of loans

Year	INF	NCAP	TN	NTA	INT	NGDP	UR	EC	ADV
1989	17.4	68.790	437	57.910	10.9	21.946	12.0	0	0
1990	9.0	63.110	314	134.826	13.7	27.597	11.8	0	0
1991	20.7	172.500	316	145.758	12.7	32.223	11.1	0	0
1992	24.4	803.000	146	64.372	13.4	34.586	13.1	0	0
1993	22.9	542.040	39	38.153	16.4	42.367	11.7	1	0
1994	35.2	400.921	62	16.822	16.4	40.895	12.5	1	0
1995	49.4	268.354	153	51.501	16.7	38.980	10.2	0	0
1996	23.2	217.820	239	153.185	17.6	42.262	9.1	0	0
1997	17.3	185.694	168	16.176	17.2	42.460	12.1	0	0
1998	18.1	157.235	153	21.079	17.5	40.431	13.1	0	0
1999	20.1	130.920	412	106.843	17.1	44.659	13.5	0	0
2000	12.8	146.289	1096	182.257	17.8	52.634	14.3	0	1
2001	11.4	133.279	933	210.294	17.4	54.676	14.2	0	1
2002	15.8	115.094	1081	323.124	16.9	65.082	12.8	0	1
2003	15.6	99.563	906	838.617	16.5	67.354	11.0	0	1
2004	15.2	263.909	2080	262.439	16.2	70.129	14.6	0	1

In the next step of the proposed methodology, we apply the genetic Algorithm to find the best combination of the functions of the independent variables for NTA in which all coefficients are significant. This algorithm results in a model given by Eq. 5 in which SSE = 136234.9, the coefficient of multiple determination of $R^2 = 0.78$ and p-value = 0.00138.

$$NTA=429.54+9.03NDDP-2.834UR^2+230.78ADV-1.96INT^2 \tag{5}$$

Figure 4 shows both the observed and the estimated NTA values in different years. Series 1 and 2 in Fig. 4 show the observed and the estimated values of NTA, respectively.

In addition, the best combination of the functions of the independent variables for TND in which all coefficients are significant with SSE of 455612.3, $R^2 = 0.90$ and a p-value of 0.00002 was:

$$TND = 79.34\sqrt{NCAP} + 931.02ADV - 895.65EC + 4.26UR^2 - 85.2INT \tag{6}$$

Figure 5 shows the observed and the estimated values of TND, in a similar manner as in Fig. 4.

The next step in the proposed methodology is to employ the result of the GA in the designed neural network. Thus, the input vector of the neural network used for NTA contains A, NGDP, UR^2, ADV and INT^2 variables and $\sqrt{NCAP}$ ADV, EC, UR^2 and INT for TND.

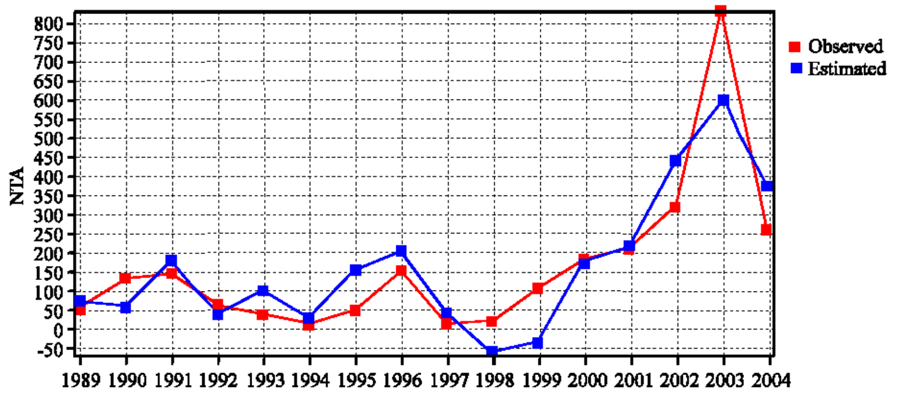

Fig. 4: Observed and estimated NTA in different years from the Genetic Algorithm model

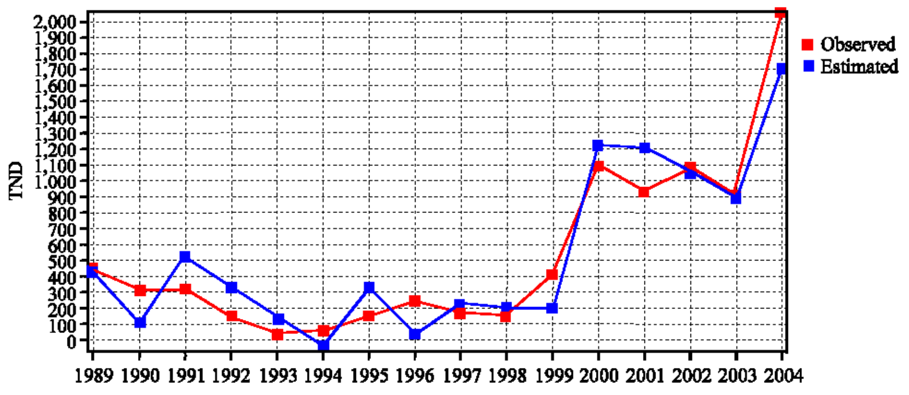

Fig. 5: Observed and estimated TND in different years (The red diagram shows observed TND and the blue one shows estimated TND from Genetic Algorithm model)

These input vectors were then transferred by a simple linear function to [0, 1] interval to ease the learning process of the two networks; one for NTA and the other for TND.

In the learning process of the network for NTA, we applied the leave-one-out method and found that the best results were obtained by using a BPN with one hidden layer of five nodes which was trained in 1250 epochs with temperature rate of 0.15 and the learning rate of 0.15. The corresponding SSE of the trained neural network obtained as 128651.3, which was 5.57% less than the corresponding SSE of the model in which the Genetic Algorithm was applied alone. Figure 6 shows the observed and the estimated net total amount of loans in different years.

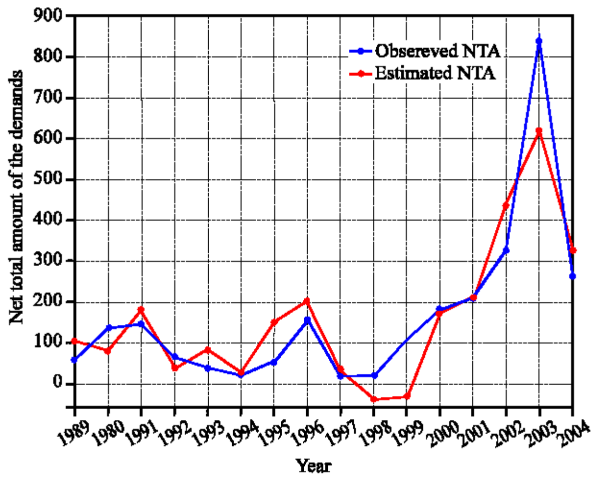

Fig. 6: Observed and estimated NTA in different years (from the introduced hybrid method)

In order to compare the performance of the hybrid algorithm with a simple neural network model, we assume that the input vector contains a simple function of the independent variables as A and NGDP, UR, ADV and INT. The designed neural network, which we call simple, was trained by the same conditions as of the hybrid method. We trained both networks (simple and hybrid) with different number of epochs and calculated their corresponding sum of squared errors.

Table 2: SSEs of the simple and the hybrid neural network for different epochs in NTA case

No. of epochs	1250	1000	800	700	600	500
Hybrid algorithm's SSE	128651.3	130383.8	133832.1	135712.9	141683.6	151802.3
Simple Neural Network's SSE	131032.5	132832.3	136063.8	138913.7	146572.1	156274.4
Difference	2381.2	2448.5	2231.7	3200.8	4888.5	4472.1

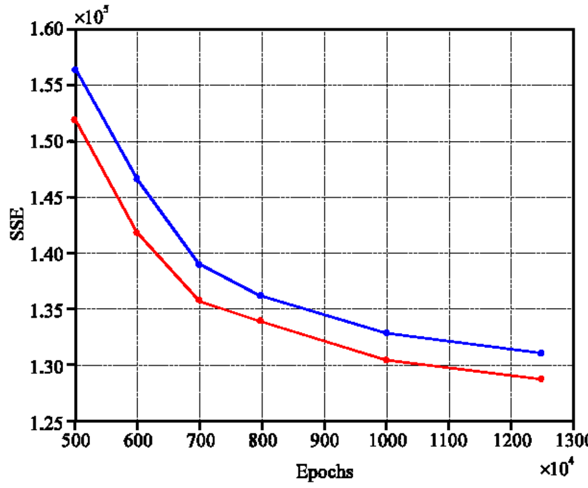

Fig. 7: SSE of the simple (blue diagram) and the hybrid algorithm (red diagram) for different epochs in NTA case

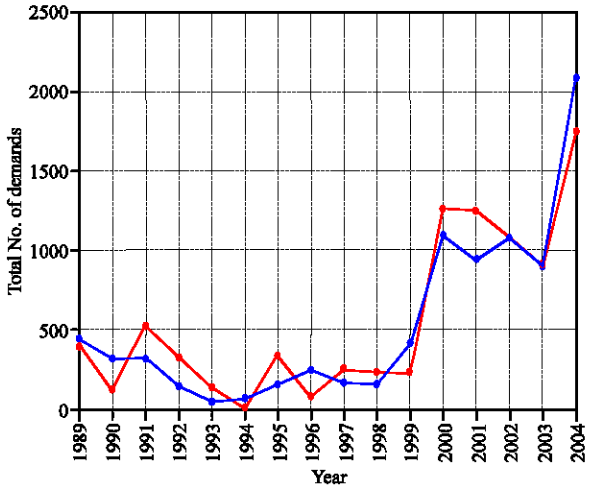

Fig. 8: TND in different years (The blue diagram shows observed TND and the red one shows estimated TND from the introduced hybrid method)

It should be mentioned that since the initial weight matrix in the neural networks is random, for any specific number of epochs, several networks were trained and the average result of each part was used. For example, in order to obtain the results in Table 2, for each number of epochs, we trained the network 100 times and the average SSE was entered in the Table 2.

The results of either Table 2 or Fig. 7 show that on average the SSE of the hybrid algorithm is 2.31% less (better) than the simple neural network modeling. Moreover, the p-value of a paired t-test for the difference between the mean SSE of the methods obtained less than 0.0005, indicating the hybrid method statistically performs better than the simple neural network. Since the SSE of the hybrid method was 5.57% better than the genetic algorithm's SSE in the NTA case, we may conclude that in this case the proposed hybrid algorithm performs better than both the genetic algorithm and the neural network when they are solely used.

We performed a similar study for the TND case in which the input vector was obtained by the genetic algorithm. Hence, the input vector of the neural network section of the hybrid algorithm in TND case contains variables $\sqrt{\text{NCAP}}$, ADV, EC, UR^2 and INT.

Once again the input vector of the network was transferred by a simple linear function to [0,1] interval to ease the learning process of the designed neural network.

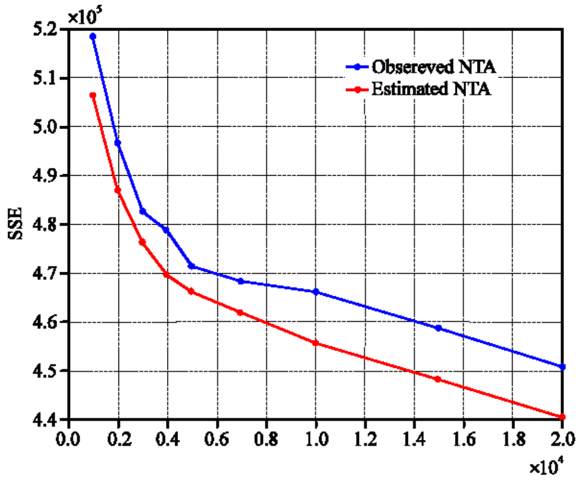

Fig. 9: SSE of the hybrid algorithm (red diagram) and the simple neural network (blue diagram) for different epochs in TND case

Then, we applied the leave-one-out method to train the BPN. A network with one hidden layer and four nodes, which was trained by 20000 epochs with temperature rate of 0.1 and the learning rate of 0.05, resulted in SSE of 440442.2. This value is 3.32% better than the SSE of the genetic algorithm itself (Fig. 8).

For the comparison study, in TND case we did the same as for the NTA case. The SSEs of both the simple and the hybrid methods are shown in Table 3 and Fig. 9.

Table 3: SSEs of the hybrid and the simple neural network methods in TND case

No. of epochs	Hybrid algorithm's SSE	Simple neural network's SSE	Difference
20000	440442.2	450609.6	10167.4
15000	448142.1	458612.9	10470.8
10000	455612.3	465981.1	10368.8
7000	461774.9	468087.2	6312.3
5000	465973.6	471322.6	5349.0
4000	469696.9	478755.1	9058.2
3000	476253.1	482412.7	6159.6
2000	486992.7	496432.0	9439.3
1000	506239.5	518402.2	12162.7

The results of Table 3 shows that on average the SSE from the proposed hybrid method is 1.85% less (i.e., better) than the SSE from the simple neural network method. In addition, the p-value of a paired t-test is almost zero, indicating that the mean SSE of the hybrid method is statistically less than the one from the simple network. Since the SSE of the hybrid method is also 3.32% better than the SSE of the genetic algorithm we may conclude that using the hybrid method results better than the cases where the neural network or the genetic algorithm is solely used.

CONCLUSION

In this study, we proposed a hybrid neural network-genetic algorithm method for econometrical modelling and data analysis. In this method, first different observed vectors and different functional forms of each vector were introduced as input to the genetic algorithm part of the method. The output of the GA is the best linear combination of the functional forms that is statistically reliable. This combination then was input to a neural network to get a more precise estimate of the mean response. The results of the case study show that when the hybrid algorithm is used, we obtain a more precise estimate of the mean response compared to those from the genetic algorithm or the neural network method solely.

REFERENCES

Ahmed, M.A. and T.M. Alkhamis, 2002. Simulation-based optimization using simulated annealing with ranking and selection. Comput. Operat. Res., 29: 387-402.

Ancona, N., L. Angelini, M. De-Tommaso, D. Marinazzo, L. Nitti, M. Pellicoro and S. Stramaglia, 2006. Measuring randomness by leave-one-out prediction error, Analysis of EEG after painful stimulation. Measuring randomness by leave-one-out prediction error, Analysis of EEG after painful stimulation. Physica A: Stat. Mech. Appl., 365: 491-498.

Angelis, L., E. Bora-Senta and C. Moyssiadis, 2001. Optimal exact experimental designs with correlated errors through a simulated annealing algorithm. Comput. Stat. Data Anal., 37: 275-296.

Ashley, D.W., 2002. The demand for consumer credit. M.Sc. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Baffi, G., E. Martin and J. Morris, 2002. Prediction intervals for non-linear projection to latent structures regression models. Chemometr. Int. Lab. Syst., 61: 151-165.

Baragona, R., F. Battaglia and C. Calzini, 2001. Genetic algorithms for the identification of additive and innovation outliers in time series. Comput. Stat. Data Anal., 37: 1-12.

Bishop, C., 1995. Neural Networks for Pattern Recognition. 1st Edn. Oxford University Press, UK.

Cawley, G.C. and N.L.C. Talbot, 2003. Efficient leave-one-out cross-validation of kernel Fisher discriminant classifiers. Pattern Recog., 36: 2585-2592.

Escandón, R.Y. and A. Díaz-Bautista, 2000. A simple dynamic model of credit and aggregate demand. Department of Economics, Working Paper Series, El Colegio de la Frontera Norte.

Etheridge, H.L., R.S. Sriram and H.Y.K. Hsu, 2000. A comparison of selected artificial neural networks that help auditors evaluate client financial viability. Decision Sci., 31: 531-550.

Evans, G.R., 1997. The Budget Deficit, Department of the US Government. Academic Press, San Diego, Ca.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. 1st Edn. Addison-Wesley Longman Publishing Co. Inc., MA, USA., pp: 372.

Greene, W.H., 2000. Econometric Analysis. 4th Edn. Prentice Hall International Incorporation, Upper Saddle River, New Jersey.

Hasheminia, H. and S.T.A. Niaki, 2006. A genetic algorithm approach to find the best regression/econometric model among the candidates. Applied Math. Comput., 183: 337-349.

Haykin, S., 1994. Neural Networks; A Comprehensive Foundation. 1st Edn. MacMillan College Publishing Company, Inc., USA.

Holland, J.H., 1975. Adoption in Neural and Artificial Systems. The University of Michigan Press, Ann Arbor, Michigan.

Kilmer, R.A., A.E. Smith and L.J. Shuman, 1999. Computing confidence intervals for stochastic simulation using neural network meta-models. Comput. Ind. Eng., 36: 391-407.

Kim, K.J. and I. Han, 2000. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. Exp. Syst. Applied, 19: 125-132.

Lee, D.S., V.S. Vassiliadis and J.M. Park, 2004. A novel threshold accepting meta-heuristic for the job shop scheduling problem. Comput. Operat. Res., 31: 2199-2213.

Lippmann, R.P., 1987. An introduction to computing with neural networks. IEEE Acoustics, Speech and Signal Processing Magazine, 4: 5-22.

Man, K.F., K.S. Tang, S. Kwong and W.A. Halang, 1997. Genetic Algorithms for Control and Signal Processing. Springer Verlag, London.

Neter, J., M.H. Kutner, C.J. Nachtsheim and W. Wasserman, 1996. Applied Linear Statistical Models. 4th Edn. McGraw Hill, Boston, Massachusetts.

Patterson, D., 1996. Artificial Neural Networks. 1st Edn. Prentice Hall, Singapore.

Ricals, I. and L. Personnaz, 2000. Construction of confidence intervals for neural networks based on least squares estimation. Neural Network, 13: 463-484.

Steil, J.J., 2006. Online stability of backpropagation-decorrelation recurrent learning. Neurocomputing, 69: 642-650.

Vera, L.V., 2002. The demand for bank loans in Venezuela: A multivariate co-integration analysis. Investigación Económica, XLII: 1-28.

Winker, P. and M. Gilli, 2004. Applications of optimization heuristics to estimation and modeling problems. Comput. Stat. Data Anal., 47: 211-223.