Choosing Variables with a Genetic Algorithm for Econometric models based on Neural Networks learning and adaptation.

Daniel Ramírez A., Israel Truijillo E.
LINDA LAB, Computer Department, UNAM
Facultad de Ingeniería
CU, México
daniel@grupolinda.org

Juan M. Gómez G
Control Department, UNAM
Facultad de Ingeniería
CU, México
juan@control.fi-b.unam.mx

Abstract:
The mixture of two already known soft computing techniques, like Genetic Algorithms and Neural Networks (NN) in Financial modeling, takes a new approach in the search for the best variables involving an Econometric model using a Neural Network. This new approach helps to recognize the importance of an economic variable among different variables regarding econometric modeling. A Genetic algorithm constructs a set of working neural networks, evolving the inputs given to each NN as well as its internal architecture. An input subset is chosen by the genetic algorithm from a multiple variable set, due to the NN training results from this given input. At the end of the evolutionary process, the best given inputs for a specific neural network architecture are obtained. The experimental results revealed an improvement of 80% in the NN learning performance of the Econometric model. At the same time it reduces the model complexity by 46%, without large computer resources being used during the evolutionary process.

1. Introduction

The complexity of econometric models and the number of variables that can be involved interfere with the creation of models and with the search for automatic tools for generating predictions. This problem has been tried to be approached for many years. The use of softcomputing techniques, such as neural network and genetic algorithms, serve to develop tools in the prediction of certain currency or in a time series analysis [1,2,3,4]. In the same research area we can also find another tendency with the use of evolutionary processes. These processes serve to modify the structure of neural networks predictors[5].

We can also find a constant in all the problems regarding time series analysis. In such area, one of the most common problems is a correct selection of the variables that are to be used. In this election, one searches for the minimum, useful and sufficient groups of variables to maintain the information relevant to the system. The contained data correlation must be small and it must contain independency against each variable. Other approaches have been done using neural networks and their inner weights modification in the search for choosing variables [9].

More conventional methods, such as principal component analysis (PCA), are effective for reducing the space and the variable quantity. Other method for measuring the energy and the quantity of noise in the signals is Fourier Analysis. These methods will be used further more in this paper in a comparative analysis against the method here presented.

2. System Overview

The system consists of an evolutionary algorithm which modifies the configuration of an artificial neural network population. These networks are a Feedforward type (NNFF) and are trained with the use of a typical backpropagation algorithm.

The input assigned to each neural network is modified using the same genetic algorithm since each individual of the population has an input mask assigned. This mask indicates which variable will be used in each NN. These variables will be selected from a larger set of variables. Such mask will be evolved using the traditional genetic function. The variable selection properties [10] embedded in the neural networks will be used to analyze the contribution from each variable.

The testing experimentations were made using a 10 variable data set, being these variables a selection of financial [11, 12] and noisy data (fig 1.)

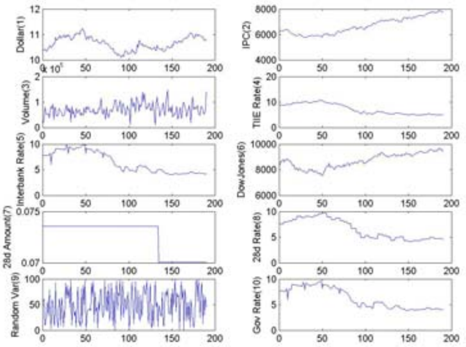


Fig. 1 Variables

3. Evolution Procedure

During the evolutionary process, the research was only around the population matrix, in which the NN and their associated masks are codified as shown in Fig 2. The population matrix was affected by applying the typical mutation, Crossover and Selection [8] genetic functions over each population individual. The NN codification was made as follows:

(input_mask,layers,neurons1,flayer1,neurons2,flayer2,

…,neuronsN,flayerN,epochs1,error1,epochs2,error2)

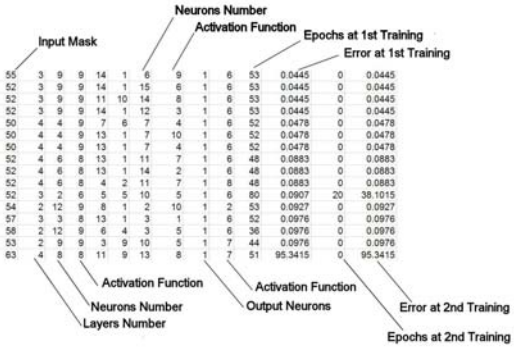
"Layers" indicate the number of layers, "neuronsN" indicate the amount of neurons in the N layers, and "FlayerN" indicates the activation function in each layer.

The genetic algorithm structure here used was a (m+l) type in which the parents compete with their offspring in the next generation. The software developed for this research allows a (m,l) genetic algorithm option, but no test was done with this modifier.

The tested population size was 75,100,125 individuals, and evolved for 75 and 100 generations. The genetic operators were applied in a 0.6, 0.3 and 0.005 rate assigned for selection, crossover and mutation respectively.

3.1. Network Evolution

The size and shape of the neural network represented in the population matrix are modified in their number of layers, number of neurons per layer and the activation function within each layer. All these variables have a maximum range to avoid excessively grown networks. The evolution process takes place by applying the genetic operator to each and every vector within the population matrix (fig 2).


Input Mask
Neurons Number
Activation Function
Epochs at 1st Training
Error at 1st Training
Activation Function
Neurons Number
Layers Number
Activation Function
Output Neurons
Error at 2nd Training
Epochs at 2nd Training

55	3	9	9	14	1	6	9	1	6	53	0.0445	0	0.0445
52	3	9	9	14	1	15	6	1	6	53	0.0445	0	0.0445
52	3	9	9	11	10	14	8	1	6	53	0.0445	0	0.0445
52	3	9	9	14	1	12	3	1	6	53	0.0445	0	0.0445
50	4	4	9	7	6	7	4	1	6	52	0.0478	0	0.0478
50	4	4	9	13	1	7	10	1	6	52	0.0478	0	0.0478
50	4	4	9	13	1	7	4	1	6	52	0.0478	0	0.0478
52	4	6	8	13	1	11	7	1	6	48	0.0883	0	0.0883
52	4	6	8	13	1	14	2	1	6	48	0.0883	0	0.0883
52	4	6	8	4	2	11	7	1	8	48	0.0883	0	0.0883
52	3	2	6	5	5	10	5	1	6	80	0.0907	20	38.1015
54	2	12	9	8	1	2	10	1	2	53	0.0927	0	0.0927
57	3	3	8	13	1	3	1	1	6	52	0.0976	0	0.0976
58	2	12	9	6	4	3	5	1	6	36	0.0976	0	0.0976
53	2	9	9	3	9	10	5	1	7	44	0.0976	0	0.0976
63	4	8	8	11	9	13	8	1	7	51	95.3415	0	95.3415

Fig. 2 Population Matrix

3.2. Mask Evolution and Definition

The input mask is defined by a binary code string whose length equals the amount of variable within the data set. The state of each bit in the mask indicates whether the variable represented by the bit will be used or not as a neural network input. All the variables assigned by the input mask will be used in the NN training therefore causing a NN weights modification resulting in an output error variation. All these parameters collaborate in the population performance dynamics, therefore a NN with an input variable that is less related to the output will have a poorer performance than a neural network with a better variable assignation. This assignment causes a better NN training and generalization.

Although the input mask representation (fig. 2) is a decimal notation, the evolution process takes place by using its binary codification.

3.3. Objective Function (OF)

In the objective function definition we tried to involve all the parameters that could correlate the complexity and learning capability of the neural network. Also, keeping in mind the search for the most significant input variable is needed. The neural network generalization and input learning capability should take place, being this network as small as possible.

$$OF = ke \times error\_r^2 + \left(\frac{\max\_itemsL + items\_layers}{\max\_itemsL}\right)^2 + \left(\frac{\max\_epochs + epochs}{\max\_epochs}\right)^2 + \left(\frac{\max\_inputs + number\_inputs - \min\_inputs}{\max\_inputs}\right)^2$$

This kind of object function allows us to minimize the error as well as the parameters associated to training and complexity, as well as to select the set of input variables that are minimal and function at the same time.

With this OF, two nets with the same performance but different number of inputs will have different ranking within this function, having a better ranking the network with less number of inputs. With this process we achieve a favorable mixture causing the genetic algorithm to work with the NNs and build together a variable selection tool [10].

4. Results

The experiment with the 10 variable data set priorly defined was evolved with the parameters described in section 2. The following results were returned in which the population size growth only caused that the genetic algorithm converge faster. A generational analysis against the use of each variable was made. (Fig. 3).

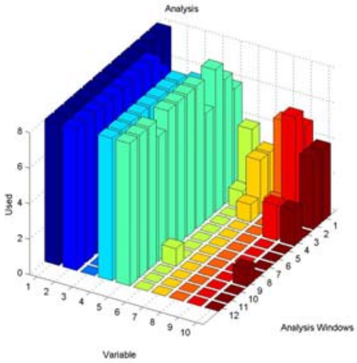


Fig. 3 Variable usage through generations

The variables selected by the genetic algorithm were 1,2,4,5 and are shown in fig 4.

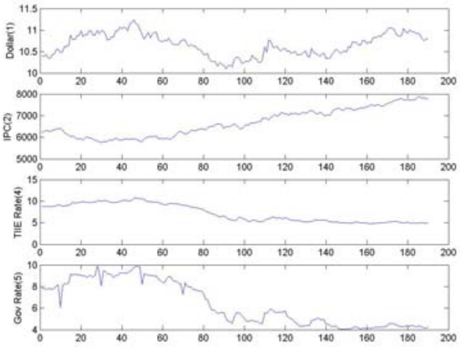


Fig. 4 Chosen Variables

The performance results with the variable selected with the genetic algorithm increased faster and we can clearly see in fig. 3 how the variables are selected and rejected through the genetic algorithm evolution. The technique also showed an increasing performance in the NNs by 80%, as shown in fig 5. Such increasement was due to a successful OF minimization. At the same time, a complexity reduction by 46% against neural networks from first generation was observed.

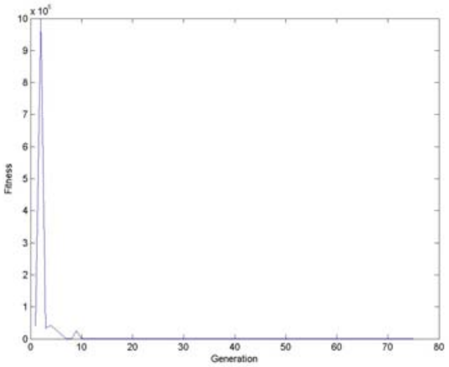


Fig. 5 Fitness minimization

5. PCA and Fourier

5.1. Fourier

The signals which are considered to be noise usually have a large amount of frequencies or, else, they have many important components in the Fourier spectrum [14]. This criteria is used to determinate the potentially random variables and eliminate them. As shown in figure 6, the variables 3 and 9 can be considered as random, using the criteria that we just explained, since these two variables have important component values in all the frequencies.

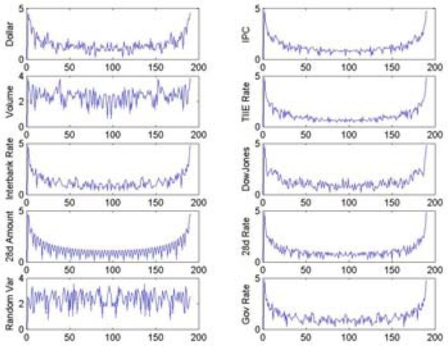


Fig. 6 Fourier Analysis

5.2. PCA

The principal component analysis (PCA) [13] is a non-parametric method, which allows us to extract relevant information from a data set that is noisy, confused and redundant. PCA is a tool to reduce the dimension of a data array with the characteristics just mentioned and to discover the hidden dynamic from the majority of the data. At the same time, this analysis allows us to find a set of orthogonal axes in which the source variables are not correlated.

5.3. Procedure

The first step in this analysis is to obtain the covariance matrix (table 1) from the variable set. This symmetric matrix contains in its diagonal the variance of each variable. If two variables are correlated, their covariance would be the same, and, if they aren't correlated at all, their covariance would be zero.

1.00	-0.11	-0.19	0.40	0.42	-0.23	-0.18	0.37	-0.02	0.42
-0.11	1.00	0.22	-0.92	-0.91	0.96	-0.80	-0.91	0.07	-0.91
-0.19	0.22	1.00	-0.25	-0.25	0.23	-0.01	-0.24	0.02	-0.25
0.40	-0.92	-0.25	1.00	0.99	-0.91	0.65	0.98	-0.08	0.99
0.42	-0.91	-0.25	0.99	1.00	-0.90	0.66	0.97	-0.06	0.99
-0.23	0.96	0.23	-0.91	-0.90	1.00	-0.71	-0.90	0.06	-0.89
-0.18	-0.80	-0.01	0.65	0.66	-0.71	1.00	0.65	-0.04	0.66
0.37	-0.91	-0.24	0.98	0.97	-0.90	0.65	1.00	-0.08	0.97
-0.02	0.07	0.02	-0.08	-0.06	0.06	-0.04	-0.08	1.00	-0.07
0.42	-0.91	-0.25	0.99	0.99	-0.89	0.66	0.97	-0.07	1.00

Table 1. Variable Covariance

The next step consists in obtaining the values and eigenvectors from the covariance matrix. The vectors are only selected when their eigenvalues result in 95% of the total. For eliminating the variables which are correlated we proceed as follows: Two variables are considered correlated when their covariance is greater than 0.6 (table 2). Then we proceed to form variables groups that are correlated. The number of main axes that remain continues to be four. The associated data to these axes is shown in figure 7. In table 2, the groups of correlated variables are also shown and named with a letter.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*Group*	
1	**1**	0	0	0	0	0	0	0	0	0	A	1
2	0	**1**	0	0	0	**1**	0	0	0	0	B	2,6
3	0	0	**1**	0	0	0	0	0	0	0	C	3
4	0	0	0	**1**	**1**	0	**1**	**1**	0	**1**	D	4,5,7,8,10
5	0	0	0	**1**	**1**	0	**1**	**1**	0	**1**	D	4,5,7,8,10
6	0	**1**	0	0	0	**1**	0	0	0	0	B	2,6
7	0	0	0	**1**	**1**	0	**1**	**1**	0	**1**	D	4,5,7,8,10
8	0	0	0	**1**	**1**	0	**1**	**1**	0	**1**	D	4,5,7,8,10
9	0	0	0	0	0	0	0	0	**1**	0	E	9
10	0	0	0	**1**	**1**	0	**1**	**1**	0	**1**	D	4,5,7,8,10

Table 2. Variable Covariance > 0.6

Then we proceed to obtain the Euler angles for each main axis as is shown in table 3. With these results we get another qualitative criteria to choose variables. If the angle of one variable is close to 45 degrees or its multiples, the variable is selected. If the angle is close to 90 degrees, the main axis is orthogonal to the variable. If the angle is close to zero degrees, the main axis is almost parallel to the variable axis.

113.97	88.40	43.72	82.36
95.05	90.53	79.32	112.23
153.28	89.55	114.73	96.18
93.62	89.91	86.19	67.06
94.57	88.89	85.58	67.11
91.98	90.96	84.94	111.95
85.22	89.26	119.47	73.19
93.96	89.81	87.22	67.32
88.90	2.61	90.60	91.88
94.37	88.98	85.67	67.11

Table 3. Dependence Analysis

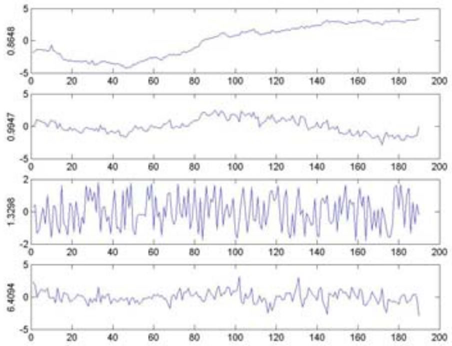


Fig. 7 Main Axes with PCA

5.4. Results

According to Fourier analysis (fig 6), the variables 3 and 9 were eliminated due to the amount of noise resulting from the quantity of energy within these signals, under a superficial inspection. From the groups that were obtained with the correlation matrix (table 1) we get the following information: Variable 1 isn't strongly correlated. However, variables 2 and 6 are correlated, causing that any of these two variables could be chosen. Variables 4,5,7,8 and 10 are as well correlated, and only one of these variables should be chosen.

The angle from the main axis obtained with PCA shows that the first main axis represents variables 4,5,7,8 and 10. The second axis is represented with 1 variable and possibly with variable 9. The fourth main axis represents variable 2.

According to this analysis, the minimum and sufficient variable set 1,2 and 4 were chosen under a criteria that is not automatic and requires expert supervision.

6. Discussion and Results

The results show resemblance in the variable assigned with the neuro-genetic method and the mix Fourier and PCA analysis. The main difference in the Neuro-Genetic analysis is independent from any human interaction. The variables aren't just chosen by their lineal independency against each other, but they are chosen due to the intrinsic characteristics in the Neuro-Genetic algorithm, which associates the information from the input variable to the output. The association is due to the way in which the input variable set affects the training and adaptation of the NNs.

The principal component analysis reduces the amount of variables and delivers us a transformation matrix. With this transformation we can reduce the number of inputs, but we still need to apply the full data set.

This analysis, as well as the Fourier energy and noise analysis, are still qualitative ways to choose an adequate variable set, and further work is needed to make them automatic. Regarding the computational time and cost, the Neuro-Genetic algorithm to choose variables requires a much greater amount of computational time, since a full evolution takes around 3 hours. The time performance and the processor load analysis are not discussed in this paper since this element wasn't considered in the analysis.

7. Conclusion

The evolution process of the FFNN configuration trained with Backpropagation and the modification of the variable input mask that is associated with each NN showed to be a good performance method regarding the variable selection in a time series modeling problem. The selection kept away the variables that were needed and the variables that had high noise factor. The process allows us to analyze the use of the variables and their performance through the evolution process.

8. References:

[1] Norio Baba, et al. Utilization of Soft Computing Techniques for Constructing Reliable Decision support Systems for Dealing Stocks, IEEE 2002

[2] Mohd. Haris Lye b. Abdullah, V. Ganapathy, Neural Network Ensamble For Financial Trend Prediction, 2000

[3] C. Haefke, C. Helmenstein, A Neural Network Model to Exploit the Econometric Properties of Austrian IPOs, Computational Intelligence for Financial Engineering, 1995, Proceeding of the IEEE/IAFE 1995, 9-15, April 1995.

[4] D. Ormoneit, R. Neuneir, Experiments in Predicting the German Stock Index DAX with Density Estimating Neural Networks.

[5] Ying Liu, Xio Yao, Evolving Neural Network for Hang Seng Stock Index Forecast, Evolutionary Computation 2001, 27-30 May 2001.

[6] M. A. Kaboudan, Genetic Evolution of Regression Models for Business and Economic Forecasting, 1999 IEEE

[7] Shu-Heng Chen; Chun-Fen Lu, Would evolutionary computation help in designs of ANNs in forecasting foreign exchange rates?, Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on , Volume: 1, 6-9 July 1999, Pages: 274 Vol. 1

[8] Goldberg David E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company, Inc., 1989.

[9] Bossaert, F.; Benjamin, D., AINS: architecture independent neuron selection, Neural Networks, 1999. IJCNN '99. International Joint Conference on , Volume: 3 , 10-16 July 1999, Pages:1866 - 1869 vol.3

[10] Y. Bennai and F. Bosseart. A neural network based variable selector. Intelligent Engineering Systems Through Artificial Neural Networks, 5, 1995 Edited by C.H. Dagli et al., Asme Press.

[11] Banco de México, Economic and Financial Indicators, Historical Data from 2003, http://www.banxico.gob.mx/

[12] DownloadQuotes, Historical Index Data, Dow Jones 2003, http://www.downloadquotes.com/en/

[13] Jolliffe, Ian T. Principal Component Analysis, New York : Springer, 2002

[14] Mix, Dwight F., Random Signal Processing, Prentice Hall, Inc., 1995.