Genetic Algorithm for Designing Artificial Neural Networks in Time Series Forecasting with Intrinsic Mode Functions

V. Landassuri-Moreno, J. Figueroa-Nazuno

Centro de Investigación en Computación
Instituto Politécnico Nacional
México DF, Col. Lindavista, C.P. 07738
victorm@sagitario.cic.ipn.mx, jfn@cic.ipn.mx

Abstract. For Time Series forecasting, we use a method called Empirical Mode Decomposition (EMD), which is adaptive and highly efficient at identifying embedded structures. EMD allows the decomposition of one-dimensional signal into intrinsic oscillatory modes. The components, called Intrinsic Mode Functions (IMF), give more information for forecasting in Artificial Neural Networks (ANNs). Genetic Algorithm (GA) is a search process based on the laws of natural selection and genetics; it has processes of *Selection, Mutation* and *Crossover* to reach a good solution. We apply GA to find the optimal structures (topologies) of ANNs; here the ANN (phenotype) is encoded to form the genome (genotype). The encoding process was made using Direct Encoding. The results show that ANNs with IMF-like inputs have better performance that ANNs with raw data-like inputs.

1 Introduction

Artificial Neural Networks have been used to forecast Time Series (TS), but the search of optimal structure takes a lot of time and sometimes we do not find a good structure that could give a better result. For this reason in this paper we use Genetic Algorithm (GA) to automatically find the optimal topology of ANNs. The phenotype is encoded as a bit string. Direct Encoding Scheme (DES) is used to encode the phenotype into genotype; DES specify in the genome every connection and node that will appear in the phenotype.

The representation of an ANN is composed of four matrices (Inputs, Bias Connect, Layer Connect, and Nodes per Layer). Each row of the matrix is concatenated, to form a bit string (a part of the genome). Then GA has four inputs, each one will be processed with his basic operations (Selection, Mutation, and Crossover). The GA has to evaluate every member of the population with one fitness function, taking every genome (every member of the population) and decoding it to form the phenotype (the four matrices that represent the ANN). When we have the phenotype, we can train the network and measure its performance. In this paper we used the Average Error (AE)- and Maximum Error (ME)-like fitness. EMD is a new nonlinear technique that gives

Avances en la ciencia de la computación en México
Research on Computing Science 13, 2005, pp. 101-110

more information on a signal. It can be a useful time series analysis tool; for this reason we apply EMD-like inputs in the ANNs to obtain a good forecasting.

The comparison in this paper is made between IMF and raw data-like inputs to the ANN designed by a GA, with AE and ME obtained from the different test sets and simulated ANN.

This paper is organized as follows. In Section 2 we describe the implementation of GA with ANNs. In Section 3 we present the EMD. In Section 4 we show the result of the computer simulation. Finally, we give the conclusions in Section 5.

2 Genetic Algorithm for Designing ANNs Topologies

GA is a method for solving optimization problems. It is based on natural selection, simulating the biological evolution. A GA at each step modifies a population. It selects individuals (parents) at random from the current population and uses them to produce children for the next generation. Since the algorithm gives preference to best parents, after several generations the population "evolves" toward an optimal solution.

To evolve ANNs we have to encode the phenotype. Here we use Direct Encoding Scheme: we specify in the genome every connection and node that will appear in the phenotype (binary matrices). Figure 1 show the cycle of the GA to design ANNs.

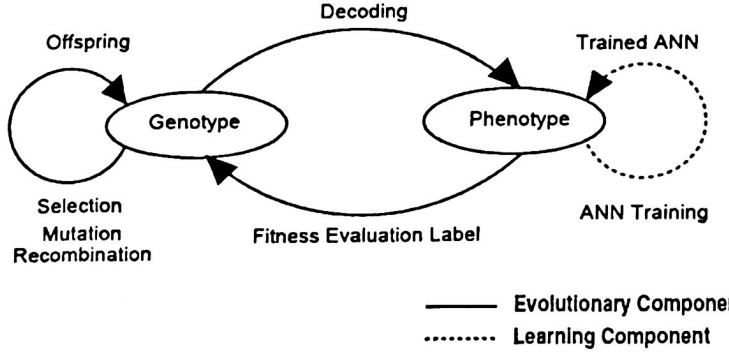

Fig. 1. Evolutionary Design of Neural Networks [3].

2.1 Parameters of the Genetic Algorithm

The GA uses three main types of rules at each step to create the next generation: *Selection* rules choose parents for the next generation basing on their fitness; *Crossover* rules combine two parents to form children for the next generation; *Mutation* rules apply random changes to individual children.

There are other functions such as the following. *Fitness scaling* converts the raw fitness scores (the value returned by the fitness function) into values in a range that is suitable for the selection function. *Reproduction* specifies how the genetic algorithm creates children for the next generation; it can be Elite fraction, which specifies the number of individuals that are guaranteed to survive to the next generation or Crossover fraction, which specifies the fraction of the next generation produced by crossover.

There are some stopping criteria to finish the GA, such as the following: Generations—to stop when the maximum number of iterations is reached; Time limit—to stop the algorithm after a certain time running. Fitness limit: if GA reaches a certain fitness value, etcetera. Our stopping criteria were set to 100 generations with 20 individual from each population; the population type is bit string, the Fitness Scaling is Rank function, the Selections is Stochastic uniform function, the Mutation is Gaussian function, and the Crossover is Scattered function.

2.2 ANN's Parameters

When the GA has to evaluate a member of the population, it calls the Fitness Function, takes the genome and decodes it in the phenotype (ANN). Then GA trains the Network to obtain the fitness value. When this is finished, the ANN is simulated with a set of data not previously seen by the ANN (test set). We used AE and ME to estimate the fitness, this value is returned to GA for continuation of the evolution. The number of epochs used in the training phase is varied because we do not know the current size of the network and if the epochs are too numerous, we will have overfitting, or otherwise, the ANN will not reach its bets performance because of too few epochs.

The mean square error (MSE) of the ANN has one problem: in case of over-fitting the MSE is close to zero, and the forecasting will be wrong, because the network learned the training set. For this reason we decided to use the Average Error and Maximum Error as others fitness measures. The training algorithm for the ANNs is the Levenberg-Marquardt algorithm; with Linear Transfer Functions in output layers and Tan-Sigmoid in hidden layers. The genome consists of four variables representing Inputs, Bias Connections, Layer Connections, and Nodes per Layer; with this we can build the ANN to train it.

2.3 Encode & Decode

Our phenotype consists of four matrices, which represent the network (Direct Encoding Scheme); all lines of a matrix are concatenated to form variables of the genome, see Fig. 2.

The first matrix of the phenotype is the Inputs, see Fig. 2, where its size is Minput (layers by number of input variables). The second is Bias Connections that represents whether or not the layer has a bias, Mbiasconnect (layers by 1). The third is Layer Connections that represent the connection between nodes Mlayerconnect (layer by layer). The fourth is the Nodes per Layer Mnode/layer (layer-1 by 3); 3 bits to permit

maximum of seven nodes per layer. The lines of these four matrices are concatenated, each one representing one variable. These are introduced as the genome of GA. In the matrix Mnode/layer we put three bits to restrict the size of the network; its size is layer-1 by 3.

Theoretical works which show that a single hidden layer is sufficient for ANNs to approximate any complex nonlinear function with any desired accuracy [6, 8]. Most authors use only one hidden layer to forecast. However, one-hidden-layer networks may require a very large number of hidden nodes, which is not desirable since the training time and the network generalization ability will degrade. Two-hidden-layer networks may provide more benefits for some type of problems [1]. Several authors address this problem and consider more than one hidden layer (usually two hidden layers) [16]. Here we permit more hidden layers.

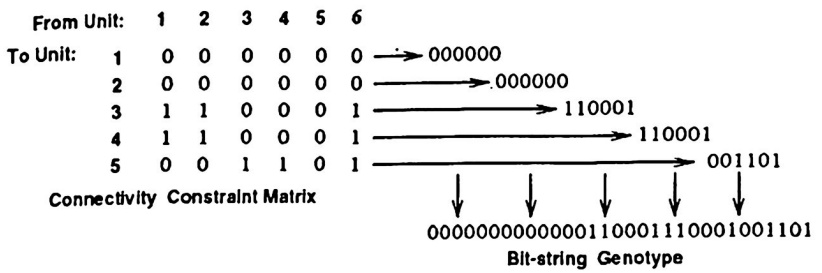

Fig. 2. Encoding of genotype from phenotype (Input matrix). The 1's represent connections between inputs variables to nodes, e.g., (3, 1) stands for connection from variable 1 to node 3, and (3, 6) for connection form variable 6 to node 3.

3 Empirical Mode Decomposition

The Empirical Mode Decomposition was introduced by Huang *et al.* in 1998 [12]. The idea of this technique is to divide a signal $x(t)$ into a sum of functions satisfying an assumption: they have at least two extrema, one maximum and one minimum. The method essentially involves tow steps:

1. Two smooth splines are constructed connecting all the maxima of $X(t)$, respectively, to get its upper envelope $X_{max}(t)$ and $X_{min}(t)$. The extrema are found by determining the change of sign of the derivative of the signal. All the data points should be covered by the upper and lower envelopes.
2. The mean of the two envelopes is subtracted from the data to get a difference signal $X_1(t)$,

$$X_1(t) = X(t) - \frac{X_{max}(t) + X_{min}(t)}{2} \tag{1}$$

While it satisfies the criteria of an intrinsic mode function, this process is repeated. The first IMF $C_1(t)$ is obtained after a certain number of iterations limited for standard

deviation (SD), computed for two consecutives shifting results. In our experiments SD is set to 0.3. The original signal can be reconstructed using the following equation where r_n is the residue.

$$X(t) = \sum_{j=1}^{n} C_j(t) + r_n(t) \tag{2}$$

Another way to explain how the empirical mode decomposition works is as follows: it picks up the highest frequency oscillation that remains in the signal. Thus, locally, each IMF contains lower frequency oscillations than the one extracted before. This property can be very useful to pick up frequency changes; since a change will appear even more clearly at the level of an IMF [10]. Fig. 3 shows one signal with its eight IMFs.

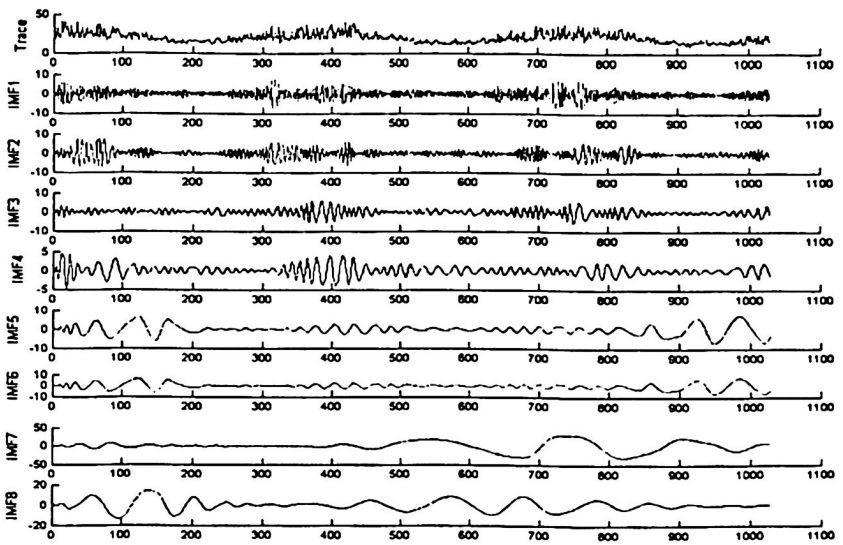

Fig. 3. Daily Maximum Temperatures in Melbourne, original data and IMF 1–8 from Australia, 1981-1990. (Source: Australian Bureau of Meteorology.)

4 Results

All ANNs of the GA were trained to forecast one step ahead. We used some Time Series to prove the performance of GA to design ANNs with IMF and without it (raw data). The Time Series were:

Meteorology: Daily Maximum Temperatures in Melbourne (DMaxTM); Daily Minimum Temperatures in Melbourne (DMinTM); Daily Precipitation, Hveravellir (DPH); *Micro-Economic*: Daily Morning Gold Prices (DMGP). *Hydrological*: Annual Minimum Level of Niel River (AMinLNR); Water Level of Corpus Christi, Texas, Estuary (WLCC).

All the time series have one variable, the only case is WLCC that have six variables (pressure, wind speed, wind direction, significant wave height, Dominant wave period and mean wave directions). This time series forecasting has the best performance, with smallest AE and ME.

In some cases with the AE, the performance of the network was very good, but the Series forecast had many jumps, crossing the real data (test set); this causes many partial errors was zero or close to zero, and when all errors were averaged, the AE was very small with respect to all variations.

In Tables 1 and 2 we show the results obtained with AE, and in Tables 3 and 4 the results with ME, e.g., of network's format: fb[7 3 1] is a feedback network with three layers and seven nodes in the first layer, three nodes in second layer and one node in the output layer; if the network does not have the suffix "fb," this means that it is a feed-forward network.

The matrices of Layer connections are represented as in Fig. 2, where (3, 1) represents the connection of the layer 1 to 3, or (1, 3) represents the connection of layer 3 to 1 (feedback); in the following matrices, every line of the matrix is separated by semicolon. The best network obtained is marked in boldface.

Table 1. Result of simulation with IMF and Average Error.

Time Serie	AE	Network	Bias Connect	Layer Connect	Epochs
DMaxTM	2.6022	fb[6 4 1]	[1;1; 1]	[000;110;110]	42
DMinTM	1.5024	fb[2 2 1]	[1;1; 1]	[001;110;010]	60
DPH	1.9534	fb[7 7 1]	[1;1;1]	[000;110;110]	59
DMGP	3.3267	fb[4 1]	[1;1]	[01;10]	68
AMinLNR	0.2998	fb[7 1]	[1;1]	[00;11]	57
WLCC	0.0582	fb[4 3 1]	[1;0;1]	[001;100;101]	81

Table 2. Result of simulation with Raw data and Average Error.

Time Serie	AE	Network	Bias Connect	Layer Connect	Epochs
DMaxTM	4.3116	fb[7 3 1]	[1;1;1]	[001;100;010]	113
DMinTM	1.8775	fb[6 4 1]	[1;0;1]	[000;110;110]	41
DPH	2.7118	fb[5 1]	[1;1]	[01;10]	100
DMGP	2.5932	fb[7 3 1]	[1;1;0]	[001;100;010]	6
AMinLNR	0.6929	fb[7 1]	[1;1]	[01;10]	109
WLCC	0.0578	[6 1]	[1;1]	[00;10]	250

Table 3. Result of simulation with IMF data and Maximum Error.

Time Serie	ME	Network	Bias Connect	Layer Connect	Epochs
DMaxTM	8.3562	fb[4 3 1]	[1;1;1]	[001;100;101]	11
DMinTM	3.5664	fb[6 4 1]	[1;1;1]	[000;110;110]	15
DPH	17.851	[7 1]	[1;1]	[00;10]	100
DMGP	11.783	fb[4 1]	[1;1]	[01;10]	47
AMinLNR	1.3782	fb[6 4 1]	[1;1;0]	[000;110;110]	21
WLCC	0.2366	fb[4 3 1]	[1;0;1]	[001;100;101]	11

Table 4. Result of simulation with Raw data and Maximum Error.

Time Serie	ME	Network	Bias Connect	Layer Connect	Epochs
DMaxTM	15.804	[4 4 1]	[1;1;1]	[000;100;010]	100
DMinTM	5.9309	[7 1]	[1;0]	[00;10]	250
DPH	21.6	[7 1]	[1;1]	[00;10]	150
DMGP	8.9345	fb[7 6 3 1]	[1;1; 1;1]	[0000;1010; 0100;0010]	200
AMinLNR	2.3641	fb[2 2 1]	[1;1;1]	[001;110;010]	17
WLCC	0.3227	fb[2 2 1]	[1;1]	[001;110;010]	50

Next, we presented the results obtained for the Annual Minimum level of Niel River. As can be observed, without IMF the network cannot give a good forecasting, see Fig. 4. Here IMF and AE slightly outperform Fig. 5. In fig. 6 the best forecasting obtained with IMF and ME is shown.

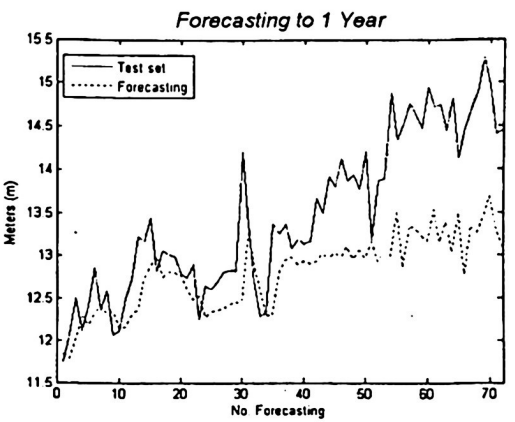

Fig. 4. Forecasting with raw data and AE = 0.6929 (1 step ahead). Annual Minimum level of Niel River.

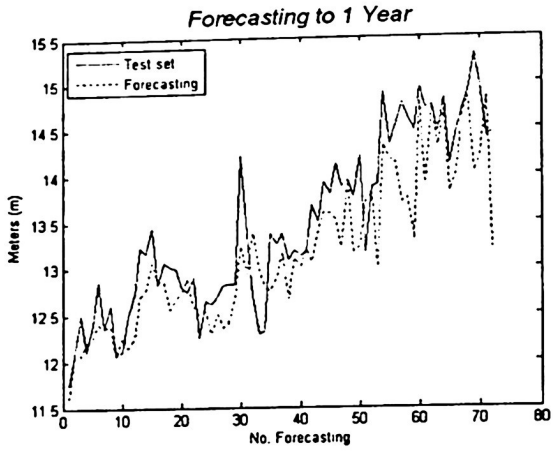

Fig. 5. Forecasting with IMF data and AE = 0.2998 (1 step ahead). Annual Minimum level of Niel River.

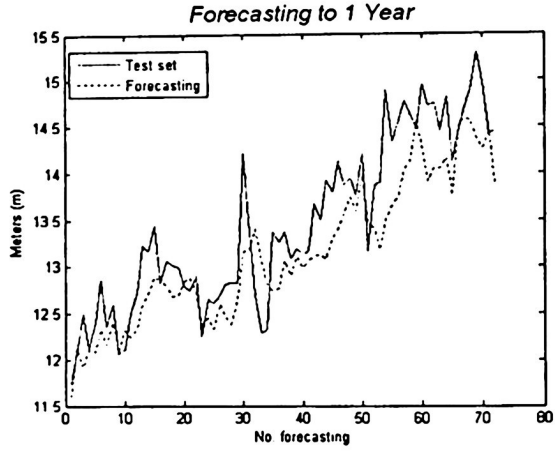

Fig. 6. Forecasting with IMF data and ME = 1.3782 (1 step ahead). The better forecasting for Annual Minimum level of Niel River.

Fig. 7 presents the best forecasting for WLCC whit raw data and AE, the experiment with IMF and ME are too similar, but the complexity of the network was very high. Thus we consider the best the data of the Fig 7.

We can see four TS with IMF, which give the best performance (DMaxTM, DMinTM, DPH, and AMinLNR) and the others two, with raw data (WLCC and DMGP). However, we can consider that WLCC is better with IMF, too, because the results are very similar for IMF and raw data. In general IMF gives good results.

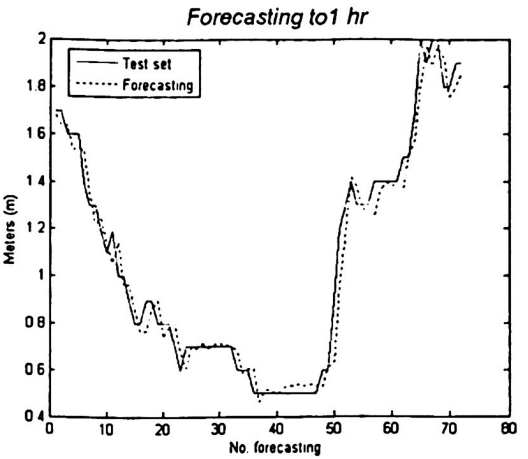

Fig. 7. The better forecasting with Raw data and AE = 0.0578 (1 step ahead). Water Level for the Corpus Christi, Texas, Estuary.

5 Conclusions

As our results show, IMF data as input to the network is better than raw data. However, as in Water level of Corpus Christy (WLCC) example, the network with raw data and AE provides the best forecasting. If we have a sufficient number of variables that affect, directly or indirectly, the phenomena that we want to predict, then we will have a better forecasting. The only problem is the following: when the signal is divided in its Intrinsic Mode Functions, the variables are too numerous for the network. As in WLCC with IMF and ME, the forecasting is very similar, but the complexity of the network was larger than the network with raw data and AE.

In DMGP the better forecast was obtained by the network with raw data and ME, comparing with the network of IMF and AE. It has bad predictions in the last points. We conclude that dynamic time series can affect the networks and become a problem to find a good structure with GA.

The networks obtained by the GA were training with the same test data with 3, 6, 12, 24, 48, and 72 steps ahead; but the error was worse than the one obtained with one step ahead. This means that the GA can find a good network for dynamic time series presented, and the network cannot learn a different dynamics. If we want time series forecasting with another step ahead, we have to run the GA again. We can apply IMF to improve forecasting if we have a small group of variables of the phenomena, but it is also convenient to use some measures of fitness. We can see in the results that GA with ANNs is a good tool to find structures, and using IMF provides better forecasting in some cases.

References

1. Barron, A.R.: A comment on "Neural networks: A review from a statistical perspective". Statistical Science 9 (1) (1994), 33–35.
2. Belew, R. Mclnerney, J., Shraudolph, N.: Evolving Networks. CSE Technical Report #CS90-174. Cognitive Computer Science Research Group June (1990), Computer Science & Engr. Dept. (C-014)
3. Blakrishnan, K., Honovar V.: Evolutionary Design of Neural Architectures – A preliminary Taxonomy and Guide to Literature. CS TR # 95-01. Department of Computer Science. Iowa State University (1995), http://citeseer.ist.psu.edu/balakrishnan95evolutionary.html.
4. Branke, J.: Evolutionary Algorithms for Neural Networks Design and Training. Technical Report No. 322, University of Karlsruhe, Institute AIFB.
5. Coughlin, K.,Tung, K. 2004a.: 11-year solar Cycle in the stratosphere extracted by the empirical mode decomposition method. Adv. Space Res., 34, 323-329.
6. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematical Control Signals Systems 2 (1989), 303–314
7. Flandrin, P., Rilling and Goncalves, P.: Empirical Mode Decomposition as a filter bank. IEEE Sig. Proc Lett., (2003)
8. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2 (1989), 359–366
9. Kuri, A.: A Comprehensive Approach to Genetic Algorithms in Optimization and Learning, Theory and Applications. Vol 1. Fundations, Coleccion de Ciencia de la Computación, Centro de Investigación en Computación, Instituto Politécnico Nacional. México (1999)
10. Magrin, I., Baraniuk, R.: Empiricla mode descomposition based time–frecuency attributes, Rice University, Houston Texas (1999)
11. McCluskey, P.: Feedforward and Recurrent Neural Networks and Genetics Programs for Stock Market and Time Series Forecasting. Department of Computer Science, Brown University. CS-93-36
12. N. Huang., Shen, Z., Long, S., Wu M., Shih H., Zheng Q., Yen N., Tung C., and Liu H.: The empirical mode decomposition and hilbert spectrum for nonlinear and non-stationary time series analysis. *Proc. of the Royal Society of London* vol. 454 (1998), pp. 903–995
13. Stanley, K., & Miikkulainen, R.: Efficient Evolution of Neural Networks Topologies. Proceedings of the 2002 Congress on Evolutionary Computation (CEC '02). Honolulu, Hawaii: IEEE
14. Yao, X., & Liu, Y.A New Evolutionary System for Evolving Artificial Neural Networks, IEEE Transactions on Neural Networks, 8(3):694-713, May (1997)
15. Yao, X. A Review of Evolutionary Artificial Neural Networks. International Journal of Intelligent Systems, 8:539-567
16. Zhang, G., Patuwo, B., Hu, M.: Forecasting with artificial neural networks: The state of the art. International journal of Forecasting (1998). 14 35-62 N.H Elsiever
17. The MathWorks web site. http://www.MathWorks.com.
18. Time Series Data Library. http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/