Adaptive Polynomial Neural Networks for Times Series Forecasting

Panos Liatsis [1], Amalia Foka [2], John Yannis Goulermas [3], Lidija Mandic [4]
[1] School of Engineering and Mathematical Sciences, City University, Northampton Square, London EC1V 0HB, UK
[2] Department of Computer Science, University of Ioannina, PO Box 1186, 45110 Ioannina, Greece
[3] Department of Electrical Engineering and Electronics, University of Liverpool, Brownlow Hill, Liverpool L69 3GJ, UK
[4] University of Zagreb, Faculty of Graphic Arts, Getaldiceva 2, Zagreb, Croatia
E-mail: *p.liatsis@city.ac.uk*

Abstract - *Time series prediction involves the determination of an appropriate model, which can encapsulate the dynamics of the system, described by the sample data. Previous work has demonstrated the potential of neural networks in predicting the behaviour of complex, non-linear systems. In particular, the class of polynomial neural networks has been shown to possess universal approximation properties, while ensuring robustness to noise and missing data, good generalisation and rapid learning. In this work, a polynomial neural network is proposed, whose structure and weight values are determined with the use of evolutionary computing. The resulting networks allow an insight into the relationships underlying the input data, hence allowing a qualitative analysis of the models' performance. The approach is tested on a variety of non-linear time series data.*

Keywords - *Genetic Algorithms, Polynomial Neural Networks, Time Series, Forecasting*

1. INTRODUCTION

Neural networks (NNs) attempt to simulate the structure and functionality of the human brain. They are massively parallel networks of layers of simple interconnected units, called neurons. Neural networks are used in a variety of application areas because they are able to model complex non-linear mappings, by adapting their parameters (i.e., topology and/or weights), while demonstrating fault tolerance. A popular class of NNs is that of feedforward networks, where information flows from the input to the output layers. Within the class of feedforward networks, there is a categorisation between first-order and higher-order (or polynomial) neural networks. In essence, first-order neural networks process weighted sums of the input data, while polynomial neural networks use higher-order combinations or functions of the data, hence providing for suitable non-linear expansions of the representation space of the problem.

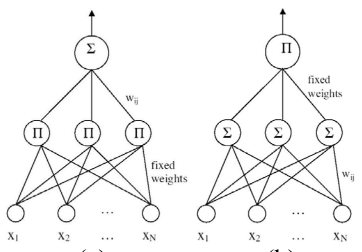
Evolutionary computing (EC) is based on processes observed in natural evolution. EC methods are based on the Darwinian principles of the survival of the fittest. Darwinian evolution is a robust search and optimisation strategy. Such approach can be applied to problems, where heuristic solutions are unavailable or simply lead to unsatisfactory results. It operates within a population of individuals, which are initially randomly selected. The individuals of a population represent the potential solutions to a particular problem. The initial population evolves towards successively better solutions by using the processes of reproduction, crossover and mutation. The fitness value of an individual gives a measure of its performance for the given problem

In this work, evolving polynomial neural networks (EPNNs) are applied to the time series prediction problem. Traditional approaches to time series prediction are based on either finding the law underlying the actual physical process or on discovering some strong empirical regularities in the observation of the time series. In the first case, if the law can be discovered and analytically described, for instance, by a set of differential equations, then by solving them, we can predict the future evolution of the time series, given that the initial conditions are known. The disadvantage of this approach is that normally only partial information is known about the dynamical process. In the second case, if the time series consists of components of periodic processes, it is possible to model it by the superposition of sinusoids. In real-world problems however, regularities such as periodicity are masked by noise, and some phenomena are described by chaotic time series, where the data seem random with no apparent periodicities.

In Section 2, we discuss some types of polynomial neural networks (PNNs), and explain the fundamental differences between traditional neural networks and PNNs. Next, the architecture of the EPNN is introduced. We describe aspects related to representation, fitness evaluation as well as the basic genetic operators. Section 4 shows the application of EPNN to the problem of time series forecasting. Finally, we draw the conclusions of this work and suggest avenues for further work.

2. POLYNOMIAL NEURAL NETWORKS

Higher-order or polynomial neural networks formulate weighted sums of products or functions of the input variables, which are then processed by the subsequent layers of the network. In essence, they expand the representational space of the neural network with non-linear terms that can facilitate the process of mapping from the input to the output space. In what follows, we will limit our discussion to the description of feedforward higher-order neural networks.



(a) (b)

Fig. 1. Architecture of (a) Sigma-Pi and (b) Pi-Sigma Networks

The sigma-pi network is a feedforward network with a single 'hidden' layer. The output of the 'hidden' layer is the product of the input terms and the output of the network is the sum of these products. They have only one layer of adaptive weights which results in fast learning. The output of the sigma-pi network is

$$y = f(w_0 + \sum_{i=1}^{h} w_i \varphi_i(v_i)) \tag{1}$$

where

$$v_i = \prod_{j=1}^{n} a_{ij} x_j$$

φ_i is the activation function at the 'hidden' layer, a_{ij} are the fixed weights and w_i are the adjustable weights.

The pi-sigma network has a very similar structure to the sigma-pi network. Their difference is that the output of the hidden layer is the sum of the inputs and the output of the hidden layer is the product of these terms. They also have a single layer of adaptive weights, however in these networks, the adaptive weights are in the first layer. The output of the network is

$$y = f(w_0 + \prod_{i=1}^{h} a_i \varphi_i(v_i)) \tag{2}$$

where

$$v_i = \sum_{j=1}^{n} w_{ij} x_j$$

with the same notation as above. Fig 1 shows the structure of sigma-pi and pi-sigma networks.

Finally, the Ridge Polynomial network (RPN) [1] is a generalisation of the pi-sigma network. It use pi-sigma networks as building blocks. The hidden layer of the network consists of pi-sigma units of varying orders and their output is summed to give the output of the network. It also has only one layer of adjustable weights in the first layer. RPNs have been shown to be universal approximators, while maintaining the fast learning properties of pi-sigma networks [2].

3. ADAPTIVE POLYNOMIAL NEURAL NETWORKS

In this work, we make use of polynomial neural networks to represent the model of the system to be identified. Each layer of the polynomial neural network is regarded as a separate optimisation problem. The input to the first layer is the independent variable of the data samples. The output of each layer is the peak nodes obtained by the use of a multi-modal GA.

The population members of the GA are network nodes represented by an eight-field bit string. The two first fields are used to represent the nodes from the previous layers, connected to the present node. The other six fields are used to represent the coefficients of a quadratic function that determines the output of node y

$$y = a + bz_1 + cz_2 + dz_1z_2 + ez_1^2 + fz_2^2 \tag{3}$$

where z_1 and z_2 are the outputs of the connected nodes in the previous layer and a, b, c, d, e, and f are the coefficients.

The fitness measure of a node is given by calculating its description length. The description length gives a trade off between the accuracy of the prediction and the complexity of the network. In this work, we use

$$I = 0.5n \log D_n^2 + 0.5m \log n \tag{4}$$

where D_n^2 is the mean square error, m is the number of coefficients in the selected model, and n is the number of observations used to determine the mean square error.

We use the multi-modal GA approach, which incorporates the fitness-sharing scheme

$$f_i' = \frac{f_i}{m_i} \tag{5}$$

where f_i is the original fitness of the node and m_i is the niche count defined by

$$m_i = \sum_{j=1}^{N} sh(d_{ij}) \tag{6}$$

with

$$sh(d_{ij}) = \begin{cases} 1-\left(\dfrac{d_{ij}}{\sigma_s}\right)^{\alpha}, & \text{if } d_{ij} < \sigma_s \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

where N is the population size, d_{ij} is the Hamming distance between the members of the population i and j, and the niche radius σ_s is given by

$$\frac{1}{2^l}\sum_{i=0}^{\sigma_s}\binom{l}{i} = \frac{1}{q} \quad (8)$$

with l being the string length and q the number of nodes in the previous network layer.

New populations are obtained after applying the genetic operators of tournament selection, single-point crossover, and point mutation. A mating restriction is also applied on the members to be crossed. If a member i is to be crossed, its mate j is selected such that $d_{ij}<\sigma_s$. If no such mate can be found, then j is selected randomly.

The procedure is repeated until the GA converges to a layer with a single node. In what follows, we will describe the specifics of the implementation of the proposed algorithm.

3.1 Representation

The potential network nodes are represented using an eight-field bit string. The first two fields represent the input of the current node, i.e., the nodes of the previous layer that it is connected with. The last six fields represent the parameters of the function used to determine the output of the current node. The size of the fields that represent the parameters of the function was set to 16 bits. The size of the fields that represent the nodes of the previous layer the current node is connected to was set according to the number of independent variables of the previous layer.

3.2 Fitness Evaluation

A node's fitness is evaluated according to its prediction error. The percentage square error (PSE) for a node is used as the node's fitness in the GA. Thus, the node's fitness is given by

$$f_i = \frac{\sum_{t=1}^{N_v}(x(t)-z_i(t))^2}{\sum_{t=1}^{N_v}x(t)^2} \quad (9)$$

where N_v is the number of points in the validation set, x(t) is the actual value of the time series at time t, and $z_i(t)$ is the output of the ith node at time t.

3.3 Selection

The search for the optimal nodes in a layer of the network is a multi-modal problem. Therefore, a niched genetic algorithm is used and the selection is performed according to the shared fitness of a node. The shared fitness is given by Eqs. (5)-(7). The constant α is set to one in order to obtain the triangular sharing function. The niche radius σ_s is determined according to a method introduced by Jelasity [3], where the niche radius is defined as a function r. This function is given by

$$r(i) = r(0)\beta^i \quad (10)$$

where i represents the layer where the search is performed, and $\beta\in(0,1)$. This function reduces at each layer, and at the final layer, the niche radius is equal to one. In our experiments, the niche radius became too small after the third layer of the network, and the GA was unable to reach the optimal solution. Hence, the niche radius is reduced only once for the second layer, and then kept constant for the subsequent layers.

The selection of members of the population to be reproduced in the following generation is performed with the tournament selection operator. In general, utilising tournament selection in combination with fitness sharing schemes has received considerable criticism. However, when used with a relatively high selection pressure (k=6), the obtained results are satisfactory. The performance of the niched GA was also tested with the use of the roulette wheel selection operator, implemented with stochastic universal sampling (SUS) as suggested in [4]. Tournament selection clearly outperformed SUS in all cases.

4. SIMULATION RESULTS

The performance of EPNN was tested with two different time series, namely the sunspots, and the Lorentz attractor. To allow a fair comparison of the results, the same number of data was used for training and validation. Therefore, in all time series, the first 2000 points are used for training, the next 500 points are used for validation and the remaining 500 points are used for testing the model with unseen data.

4.1 Sunspot Series

The first set of experiments was conducted on monthly sunspot numbers recorded by the Sunspot Index Data Center (SIDC) from January 1749 to July 1999. These numbers are indicative of the average relative number of sunspots observed every day of the month. The solar energy output of the sun ionises the particles in the ionosphere. As a result, the solar energy output determines which frequencies will pass through the ionosphere and which will bounce back to the earth. The prediction of the solar activities is therefore essential to organisations planning long-range high frequency communication links and space research related activities. The sunspot time series has been classified as quasi-periodic and it has been found that the period varies between 7 to 16 years with irregular amplitudes, making the time series hard to predict.

The objective of the experiment is to generative a single-step prediction based on past observations. The data were normalised so that the maximum value is one, before they were used as input data to the algorithms. The input pattern was set to {x(t-1), x(t-2), x(t-3)} and the desired output was x(t)=f(x(t-1), x(t-2), x(t-3)). From the 3000 available data points, 500 data points (from samples 2000 to 2500) were used for validation of the potential models. The experiments were run with a population size of 100 for 500 generations, with tournament size 6, and probabilities of crossover and mutation, 0.9 and 0.01, respectively.

The EPNN resulted to a network of four layers to model the sunspot time series, as shown in Fig. 2. The network nodes at each layer are shown in ascending order, according to their PSE. Thus nodes at the top are the ones with the smallest PSE for each layer. The most significant term in the partial descriptions,

$$y = a + bx_i + cx_j + dx_i x_j + ex_i^2 + fx_j^2 \qquad (11)$$

of the model was the term x_j and the least significant term was the constant term. The past values of the sunspot series contributed equally to the final model.

The results of the prediction are shown in Fig. 3. The percentage square error (PSE) over the entire data set is 0.057589 and the root mean square error (RMSE) is 0.004167. The PSE over the validation set is 0.052777. The difference in the PSE between the entire and the validation data sets is rather small, and thus the obtained model performs with approximately the same accuracy on both sets.

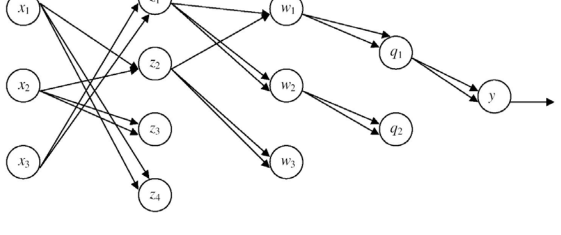


Fig. 2. Resulting Network Topology for sunspot series using the EPNN

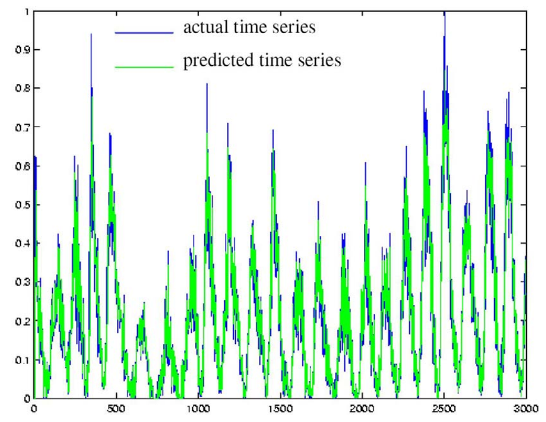


Fig. 3. Prediction of sunspot time series using EPNN

4.2 Lorentz Attractor

Lorentz proposed the Lorentz attractor system in his attempt to model how an air current rises and falls while being heated by the sun. The Lorentz attractor has also been used to model a far-infrared NH_3 laser that generates chaotic intensity fluctuations [5]. The model is given by the following set of equations

$$\begin{aligned} \dot{x} &= \sigma(x(t) - y(t)) \\ \dot{y} &= -y + x(t)(r - y(t)) \\ \dot{z} &= -bz(t) + x(t)y(t) \end{aligned} \qquad (12)$$

The time series used in our experiments is the x component of the Lorentz system, obtained by solving the differential equations system with initial conditions σ=10, r=50 and b=8/3. Once again the data was normalised to a maximum of 1.

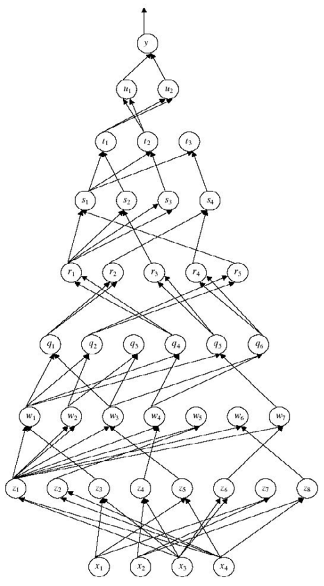


Fig. 4. Resulting Network Topologies for Lorentz series using the EPNN.

In this case, four past values, i.e., {x_1}, x(t-1), {x_2}, x(t-2), {x_3}, x(t-3) and {x_4}, x(t-4), were used for one-step prediction. The experiments were performed with 100 members in each population for 500 generations, with tournament size 6 and crossover/mutation probabilities of 0.95 and 0.03, respectively. Data points 2000 to 2500 were used for model validation. The solution reached by the EPNN had eight layers, as shown in Fig. 4. The network nodes at each layer are shown in ascending order, according to their PSE. Thus, nodes with the smallest PSE in each layer are on the left hand side, while nodes with the highest PSE are on the right hand side. The most significant term in the partial descriptions was x_i^2 and the least significant term was the constant. Variables x_3 and x_4 were the most significant variables in the model.

The results of the prediction and the actual system can be seen in Fig. 5. The PSE over the entire data set is 0.000244 and the RMSE is 0.000050. The PSE over the validation set is 0.000231. As with the sunspot time series, the difference of the PSE over the entire and the validation sets is little and thus the generalisation of the network is very good.

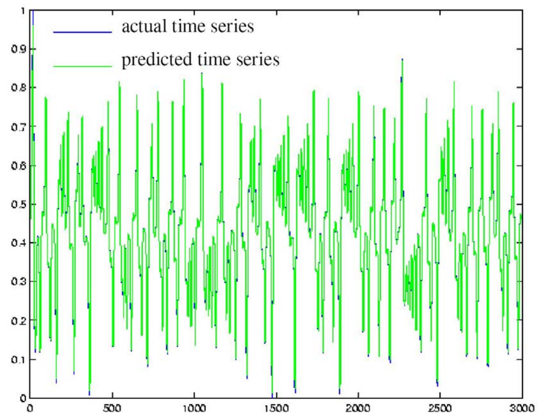


Fig. 5. Prediction of Lorentz time series with the EPNN.

5. CONCLUSIONS

This work proposed a methodology for determining the structure and weights of a polynomial neural network using Genetic Algorithms, and demonstrated its application to the problem of time series forecasting. Specific contributions of the research include the determination of the niche radius, a novel means of calculating the fitness of potential solutions, the incorporation of the elitism mechanism and the selection mechanism. The EPNN successfully combines the partial model descriptions to produce robust models that generalise very well, using only a fraction of the data points normally necessary with other models.

Further work will concentrate on improving the means for determining the niche radius. It is proposed that a co-evolutionary method, where the niche radius is evolved along with the potential solutions is used. Such a scheme was proposed in [6]. Another direction is the selection of the reproduction operator. Alternative selection schemes will be tested with the shared fitness scheme. For instance, the Restricted Competition Selection [7] has the potential to provide stable sub-populations.

REFERENCES

[1] Y. Shin and J. Ghosh, Ridge Polynomial Networks, *IEEE Trans Neural Networks*, Vol. 6, No. 3, (1995), pp. 610-622.

[2] N.R. Farnum and L.W. Stanton, *Quantitative Forecasting Methods*, PWS-KENT, 1989.

[3] M. Jelasity and J. Dombi, GAs, a Concept on Modelling Species in Genetic Algorithms, *Artificial Intelligence*, Vol. 99, No. 1, (1998), pp. 1-19.

[4] B. Sareni and L. Krahenbuhl, Fitness sharing and Niching Methods Revisited, *IEEE Trans Evolutionary Computation*, Vol. 2, No. 3, (1998), pp. 97-106.

[5] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*, Cambridge University Press, 1997.

[6] D. Quagliarella, J. Periaux, C. Poloni, C. and G. Winter, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, Wiley, 2000.

[7] C.-G. Lee, D.-H. Cho, H.-K and Jung, Niching Genetic Algorithm with Restricted Competition Selection for Multimodal Function Optimization, *IEEE Trans Magnetics*, Vol. 35, No. 3, (1999), pp. 34-53.