Forecasting Time Series with a New Architecture for Polynomial Artificial Neural Network

A. Flores-Méndez, Student Member, IEEE & E. Gómez-Ramírez, Senior Member, IEEE

Abstract—Polynomial Artificial Neural Networks have shown to be a powerful Network for forecasting non linear time series. With this type of networks it is possible to have information about the nature of the time series analyzed. However, the problem with this type of network is the computation time required and sometimes the huge number of terms of the polynomial obtained. In this paper, a novel optimization algorithm that improves the number of terms of the polynomial is presented. The architecture adaptation uses genetic algorithm to find the optimal architecture for every example. Some examples of non linear time series are shown.

I. INTRODUCTION

SEVERAL applications of Artificial Neural Networks (ANN), Fuzzy Logic and Evolutionary Algorithms have been carried out in the framework of identification. All this applications born like an alternative to forecast nonlinear dynamics like chaos or with other kind of complex dynamics. [1], [2].

For example, in adaptive control systems it is very important to find the optimal model structure to ensure the convergence of the algorithms used to obtain the controller parameters. These models in many cases require some kind of prior knowledge of the plant to be identified to define the type of model that it is going to be used. This means that it is possible to find many different combinations of algorithms or functions that represent the dynamics. However, this problem can sometimes be nonlinear and the identified model does not represent this type of behavior.

ANN are very useful to model nonlinear and chaotic dynamics [3] and there are principally two ways to improve the learning phase: preprocessing the input and adapting the architecture of the network to minimize the learning error. Some alternatives have been developed to extract structures from the data like the embedded dimension, decomposition of the measures using Wavelets [4], processing the input data or defining limits to work only in a linear region. The problem of many applications and algorithms is that they only work within the limits defined and it’s possible that they could only be applied to the example presented.

Polynomial Artificial Neural Networks (PANN) have proved their ability to model and forecast time series with a low training error compared with other types of artificial neural networks. [5]. There are some papers that describe similar approximations [6] like Higher Order Neural Networks, HONN [7]; and some others outside the ANN scope like NARMAX models [8], [9], [10], [11], Group Method of Data Handling (GMDH) [12], [13] and other kinds of polynomial approximations [14], [15]. In this paper we describe how the minimal optimal structure of PANN using Genetic Algorithm is obtained through a simple methodology that selects the terms and at the same time reduces the error. The structure uses homogeneous polynomials with integer exponents that improve the behavior of the system considering the mean square error.

The structure of the paper is the following: Section 2 describes the theory of PANN, in section 3 we outline the optimum search, section 4 introduces some concepts of Genetic Algorithm to obtain the optimal Architecture of PANN. Section 5 explains how we can use PANN to forecast nonlinear systems. In section 6 some simulations results are included to compare the performance of the theory presented and finally the conclusions of the work.

II. POLYNOMIAL ARTIFICIAL NEURAL NETWORK

The model of PANN (Figure 1) is described by [5] as:

$$\hat{y}_t = \phi(X_t, X_{k-1}, \ldots, X_{k-l}, y_{k-1}, \ldots, y_{k-m})$$

(1)

where $\hat{y}_t \in \mathbb{R}$ is the estimated output at time $k$ obtained by the nonlinear function $\phi(.)$. $X_t \in \mathbb{R}^n$ is a vector whose components are the inputs at the $k$-th time, $l$ is the number of delays of the input to be consider while $m$ is the number of delays of the output. The nonlinear function $[x]_{max}^n$, refers to an activation function defined as:

$$[x]_{max}^n = \begin{cases} x_{max} & x \geq x_{max} \\ x_{min} < x < x_{max} \\ x_{min} & x \leq x_{min} \end{cases}$$

(2)

where $x_{max}$ and $x_{min}$ are upper and lower bounds respectively.

In order to simplify the notation let us define $Z^T_k$ as follows:

$$Z^T_k := [X^T_k \ X_{k-1}^T \ \cdots \ X_{k-l}^T \  y_{k-1} \ \cdots \ y_{k-m}]$$

(3)

where $[M \ M']$ refers to the matrix obtained by
concatenating the matrices $M$ and $M'$. As an example, let $Z = \begin{bmatrix} X & Y \end{bmatrix}$, $X = \begin{bmatrix} 1 & 2 \end{bmatrix}$ and $Y = \begin{bmatrix} 3 & 4 & 5 \end{bmatrix}$ then $Z = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$. And so $Z_k \in \mathbb{R}^n$, where:

$$n_z = n(l+1) + m$$

(4)

Now let $Z^i$ stand for the set of homogeneous polynomials of total degree $i$ with its monomials defined over $Z$, and define $\Phi_p(Z^i)$ as the set of polynomials:

$$\Phi_p(Z^i) = \left\{ f(Z) : f(Z) = \sum_{i=1} \sum_{z' \in Z^i} z', \text{ where } z' \in Z^i \right\}$$

(5)

Therefore, if $\phi(Z) \in \Phi_p(Z)$ then it can be written as:

$$\phi(Z) = z^0 + z^1 + \ldots + z^p$$

$$w_0$$

$$+ w_{1,1} z_1^1 + w_{1,2} z_1^2 + \ldots + w_{1,m} z_m^1$$

$$+ w_{2,1} z_2^1 + w_{2,2} z_2^2 + \ldots + w_{2,m} z_m^1$$

$$+ \ldots + \ldots$$

$$+ w_{p,1} z_1^p + w_{p,2} z_1^{p-1} + \ldots + w_{p,m} z_m^p$$

(6)

where $w_0$ corresponds to the input bias, while $z^i$ refers to the linear terms, e.g., these terms are equivalent to an ARMAX model. The terms that have two variables are equivalent to the modulation between the inputs and the power of every one. Usually the polynomial approximations use integer exponents [5], but the relation between variables not always has this type of dynamics. As an example, a PANN with non integer powers was proposed in [16].

Now, let us define $N_i$ as the number of terms of every homogenous polynomial of degree $i$. Therefore:

$$N_0 = 1, N_i = n_z, N_2 = \sum_{z \in Z} i_i, N_j = \begin{array}{c} \sum_{z_j \in Z} i \end{array}$$

(7)

The dimension $N_{\phi_p}$ of each member of the family $\Phi_p$ can be computed as follows:

$$N_{\phi_p} = \sum_{j=0}^{N_m} N_j$$

(8)

A. Learning of PANN

To introduce the learning concepts in PANN first it is necessary to define some terms that will be used:

The approximation error of PANN can be defined as:

$$\text{err}_{r}(Y', \phi(Z)) = \frac{1}{\tau} \sum_{i=1}^{\tau} \left( y_i - \phi(Z_i) \right)$$

(9)

where $Y' = (y_1, y_2, \ldots, y_\tau)$ is the row vector whose $i$-th component is the target output at time $i$, $\phi(Z_i) \in \Phi_p(Z)$ and $\tau$ is the number of points to analyze. The optimal error is defined by:

$$\text{opt err}_{r}(Y', \phi(Z)) := \min_{\phi \in \Phi_p} \text{err}_{r}(Y', \phi(Z))$$

(10)

where $\phi(Z) \in \Phi_p(Z)$ is the optimal estimation of $Y'$.

The PANN learns uniformly the target output with accuracy $\varepsilon$ if:

$$\Pr \left\{ \text{err}_{r}(Y', \phi(Z)) > \varepsilon \right\} = 0 \quad \varepsilon > 0$$

(11)

Now the learning problem is how to find a specific $\phi'(Z) \in \Phi_p(Z)$ that fulfills the previous inequality. Let us define a vector of components $M$ in the same way as in (3):

$$M(Z) = \begin{bmatrix} 1, z_1^1, z_1^2, \ldots, z_n^1, z_1^2, z_2^2, \ldots, z_m^p, \ldots, z_1^p, z_2^p, \ldots, z_n^p \end{bmatrix}$$

(12)

It follows that the nonlinear function $\phi(Z) \in \Phi_p(Z)$ described in (6) can be represented as:

$$\phi(Z) = \langle W', M(Z) \rangle, \quad W' = W \cdot S$$

(13)

where $B = \{0,1\}, W \in \mathbb{R}^{N_{\phi_p} \times 1}$ are the weights of PANN, $S \in B^{N_{\phi_p} \times 1}$ is a boolean vector, and the product $\cdot$ is defined.
by:
\[
W \cdot S = \begin{cases} 
  w_j \text{ if } s_j = 1 \\
  0 \text{ if } s_j = 0 
\end{cases}
\]  \hspace{1cm} (14)

For example, if:
\[
W = [w_1 \ w_2 \ w_3 \ w_4]^T, \quad S = [1 \ 0 \ 1 \ 0]^T
\]

\[
W \cdot S = [w_1 \ 0 \ w_3 \ 0]^T
\]

Equation (13) means that \( \phi \) has only specific terms of \( \Phi_p \) that can be selected by \( S \) in such way, that the optimal structure of PANN \( \Phi^* \) can be calculated as:
\[
\phi(Z) = \left( (W^T) \cdot M(Z) \right) = \left( W \cdot (S)^* \cdot M(Z) \right)
\]
\[
= \left( W \cdot M(Z) \cdot (S)^* \right)
\]  \hspace{1cm} (15)

\[
\Rightarrow err_y(y^*, \phi(Z)) = err_y(y^*, \left( (W^T) \cdot M(Z) \right))
\]  \hspace{1cm} (16)

Using (9)-(16), the learning problem for this specific structure can be formulated as the following two steps optimization problem:
\[
\min_{S} \min_{n} \left( \text{err}_y \left( y^*, \phi(Z) \right) \right)
\]  \hspace{1cm} (18)

where \( \text{err}_y \left( y^*, \phi(Z) \right) \) is the error value given by (9) under a fixed value of \( S \). The values of the parameter \( W \) can be obtained using least squares method:
\[
W^*_k = \arg \min_{w \in \mathbb{R}^n} \left( \text{err}_y \left( y^*, \phi(Z) \right) \right)
\]
\[
W^*_k = \Gamma \sum_{i=1}^{n} y_i \left( M_i(Z_i) \right) \text{ where:}
\]
\[
M_i(Z_i) = M(Z_i) \cdot (S)^*, \quad \Gamma = \left( \sum_{i=1}^{n} M_i(z_i) \cdot M_i(z_i)^T \right)^{-1}
\]  \hspace{1cm} (19)

In our case \( N_p = N_{\phi_p} \), and therefore the searching space is of dimension \( 2^{N_p} \), where only a generally very small subset (in some cases compose by a unique element \( \phi(z) \)) of \( \Phi_p(Z) \), fulfills equation (17). The next section describes the method by which we search through the solution space, reducing the computational effort required to find an optimal solution. Following, we show how the value of the array \( S^* \) can be obtained using GA.

III. OPTIMUM SEARCH

As already has been stated, the solution space has dimension \( 2^{N_p} \). Therefore, we would like to find the minimum \( p \) for which \( \phi(Z) \in \Phi_p(Z) \). Moreover, the vector \( Z \) is also subject to optimization, since a number of variables (from vectors \( X \) and \( Y \)) included on it, might not be meaningful for the solution. In other words, we would like to determine the number of delays required for \( X \) and \( Y \), and once we determine these values, we only need some of their combinations. Even more, since we are dealing with polynomials, we would like to estimate first the monomials with the largest degree, and later include the less important terms for the approximation.

Now, this leads us to a tree like structure, where the depth of the node implies which problem will be tackled. Now, let \( \pi, Z \) and \( \mu \) be the maximum order, and the number of \( x \) and \( y \) delays respectively. Then, the general scheme of the method is shown in the following figure.

For this multicriteria optimization problem, we use Genetic Algorithm to solve the first three stages of the problem, and leave the last step (optimize \( \phi(Z) \)) to LMS. The following sections give a broader explanation of how we achieved this.

IV. GENETIC ALGORITHM

Genetic Algorithm (GA) is a knowledge model inspired in the evolution mechanisms observed in nature. GA usually uses the following cycle [17],[18]:

- **Generation of an initial population in a random way.**
- **Evaluation of the fitness through some objective function of every individual that belongs to the population.**
- **Generation of a new population through operations like crossover and mutation applied to best fitted individuals.**
- **Using the newly obtained population iterate until the termination criterion is fulfilled or a certain number of generations is reached.**

To find the optimal structure of the network the crossover or sexual recombination, the mutation and other special process called add parents and add Random Father [16] are used. The detailed description of these processes follows.

A. Selection Process

To explain the selection process let \( g = 1, \ldots, n_g \), where \( n_g \) is the total number of generations, let us introduce the matrix \( F_g \) which is the set of parents of a given population while \( A_g \) represents the current population. This matrices are
Boolean of dimensions \( F_g \in \mathbb{B}^{n_p \times n_i} \) and \( A_g \in \mathbb{B}^{n_p \times n_0} \) respectively, where \( n_p \) is the number of parents of the population, \( n_i \) is the number of individuals and \( n_0 \) is the size of every array (of chromosomes).

The Selection Process \( S_g \) computes the objective function \( c \) that represents the performance condition to optimize and selects the best \( n_p \) individuals of \( A_g \) as:

\[
S_g(A_g, n_p) = \min_{A_g} c(A_g)
\]

Then, the parents of the next generation can be calculated by:

\[
F_{g+1} = S_g(A_g, n_p)
\]

where the operator \( S_g \) can be used to obtain the best individual of every generation \( g \) as:

\[
S_g(A_g, 1)
\]

Crossover

Let \( C(F_g, n_i) \) be the multipoint crossover operator which can be defined as the combination of the parent’s information set considering the number of intervals \( n_i \) of each individual and the number of sons \( n_s \) such that:

\[
n_i = n_s
\]

then \( C : \mathbb{B}^{n_p \times n_i} \times \mathbb{B}^{n_s \times n_i} \rightarrow \mathbb{B}^{n_s \times n_i} \). In order to show how the crossover operator can be applied, the following example is introduced. Let \( F_g \in \mathbb{B}^{n_p \times n_i} \) (we have 2 parents) and \( n_s = 2 \).

This means that every father (array), \( a \) and \( b \), is divided in 2 sections, say \( a_i \) and \( b_i \) respectively for \( i = 1, 2 \). It is important to point out that with this operator the parents \( F_g \) of the population \( g \) are included in the result of the crossover as:

\[
F_g = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \Rightarrow C(F_g, 2) = \begin{bmatrix} a_1 & a_2 \leftrightarrow \\ b_1 & b_2 \end{bmatrix}
\]

B. Mutation

The mutation operator just changes some bits that were selected in a random way from a fixed probability factor \( P_m \); in other words, we just vary the components of some genes. This operator is extremely important, because it assures the maintenance of the diversity inside the population, which is basic for the evolution. This operator \( M : \mathbb{B}^{n_p \times n_i} \rightarrow \mathbb{B}^{n_s \times n_i} \) changes a specific population with probability \( P_m \) in the following way:

\[
M(F_g, P_m) = \begin{cases} F_g & \text{if } r(\omega) \leq P_m \\ F_g & \text{if } r(\omega) > P_m \end{cases}
\]

where \( r(\omega) \in (0,1) \) is a random variable with uniform distribution \( \omega = i, \ldots, n_i \) defined on a probability space \((\Omega, \mathcal{F}, P)\), \( \omega \in \Omega \). The mutation operator ensures that the probability of finding any point in the search space is never equal to zero. If the probability is very high the information that every generation founds can be lost and the method has the same behavior that a traditionally random search.

C. Add Parents Mutated

This process adds the mutated parents to the crossover result, and then the population at the generation \( g \), \( A_g \), is obtained as:

\[
A_g = \begin{bmatrix} C(F_g, n_i) \\ M(C(F_g, n_i)) \end{bmatrix}
\]

Notice that \( A_g \) has the best individuals of the previous generation \( A_{g-1} \), because the parents are selected in that way. This step and the previous one ensure that the algorithm does not diverge.

D. Add Random Parents

To avoid local minima a new scheme is introduced and it is called add random parents. If the best individual of one generation is the same than the previous one, a new random individual is included in the set of parents for the next generation. This step increases the population because the crossover is applied also to the new random parent, therefore the number of sons increases by the relation (23). This step is tantamount to have a very big mutation probability and to search in new points of the solution space.

E. GA applied to PANN

For our case, the application of GA to the PANN theory is automatic if we consider \( \omega = i, \ldots, n_i \), like the array searched, then the problem of learning can be reduced to obtain the optimal structure of PANN using GA. Notice that the optimal error in generation \( g \) is obtained by:

\[
\text{opt err}_i = \min_{y_i, z} \text{err}_i(\langle W \rangle_i, M(z))
\]

In summary the proposed Genetic Algorithm can be described by the following steps:

1. Let \( g = 1 \) and set \( A_0 \) to a random Boolean matrix with dimension \( A_0 \in \mathbb{B}^{n_p \times n_0} \), and set \( n_p, n_0 \) to an adequate value.
2. Obtain \( F_g = S_{g-1}(A_0, n_p) \).
3. If \( g \neq 1 \) and \( F_g = F_{g-1} \) then let \( F_g = \begin{bmatrix} F_g \\ f_r \end{bmatrix} \), where \( f_r \) represents a random father.
4. Calculate \( A_g \)
5. Let \( g = g + 1 \) and return to step 2 until the maximal number of generations \( n_g \) is reached or one of the individuals of \( S_{g-1} \) obtains the minimal desire value.

Remark: If \( \epsilon_g = \text{err}_i(\langle y_i, \langle W \rangle_g, M(Z) \rangle), i = 1, \ldots, n, \) represents the error between the target \( y \) and the estimation given by the \( i \)-th solution of the PANN, then:

8793
\[ e^*_g = \min, e^i_g = \min_{\phi_i} \left\{ y^*, \left\langle W \left| \left( M(Z) \right) \right| \right\} \right\} \]

represents the error of the optimal individual in the generation \( g \) that can be selected by \( S_g := S_g (A_{g+1}) \) due to \( S_g (A_g, n_p) = \min_{\phi} e(\phi) \).

Recall that the population \( A_g \) of generation \( g \) and the parents \( F_{g+1} \) can be described by:

\[
A_g = \begin{bmatrix} C (F_g, n_p) \\ M (C (F_g, n_p)) \end{bmatrix}, \quad F_{g+1} = S_g (A_g, n_p)
\]

\[
A_{g+1} = \begin{bmatrix} C (F_{g+1}, n_p) \\ M (C (F_{g+1}, n_p)) \end{bmatrix}, \quad F_{g+2} = S_g (A_{g+1}, n_p)
\]

Notice that the best individuals of the generation \( g \) are in the next generation due to the crossover operator, then:

\[
\min_{\phi} \left\{ y^*, \left\langle W \left| \left( M(Z) \right) \right| \right\} \right\} \leq \min_{\phi_i} \left\{ y^*, \left\langle W \left| \left( M(Z) \right) \right| \right\} \right\}
\]

\[ \Rightarrow e^*_g \leq e^i_g \]

with \( \phi_g \in \Phi_p \) described as:

\[ \phi_g = \left\{ W, M(z) \right\} \]

This means that the minimal error in generation \( g \) is less than or equal to the error of the previous generation for the best individual of the population. This condition is sufficient to prove convergence, but not to the optimal error value.

It is possible to show that if the mutation Probability \( P_m \) is strictly positive then for any small enough \( \varepsilon > 0 \) we can guarantee (29) and also:

\[ err(y, \phi_{g+1}(z)) + \varepsilon \leq err(y, \phi_g(z)) \]

**Theorem 1:** If \( S_g \) is the best individual at the generation \( g \), i.e., the optimal structure of PANN at this moment. Then the approximation error (9) can be rewritten as:

\[ err(y, \phi_g(z)) = \frac{1}{n} \sum_{i=1}^{n} \left( y_i - \left\langle W, M(z) \right\rangle \right)^2 \]

where \( y^* \) is the target output, \( \phi_g(z) \) is the optimal representation at generation \( g \), and:

\[ W_{k_g} = \arg \min_{W \in \mathbb{R}^{r \times n}} err_n \left( y^*, \phi_g(z) \right) \]

If the Mutation Probability \( P_m > 0 \) then PANN learns uniformly the target output, such that:

\[ \lim_{g \to \infty} \Pr \left[ err(y, \phi_g(z) - opterr(y, \phi(z)) \geq \varepsilon \right] = 0, \quad \varepsilon > 0 \]

V. EXPERIMENTAL RESULTS

Two examples of time series were selected to show the behavior of the algorithm. The first one is the May Equation and the second one the Lorenz Equation. The parameters used were: \( \pi = 4, \mu = 6, n = 4, p_m = 0.15, p = 2 \) (as there are no inputs), the initial population was set to 1,000 individuals, and we used \( n_g = 10 \). The objective function used is not only looks for the minimum error, but also the size of PANN (the one with less terms). In the tables shown in the following sections we include the average values for every system over 10 trials made on each system.

**A. May Equation**

The May Equation is a nonlinear equation represented by [19]:

\[ x_{n+1} = 1 - A x_n^2 \]

where \( A = 1.7 \). For the training set we used the first 150 values and the over the remaining 50 we test the system.

![Figure 3 Results of training and test for 50 samples for May Equation](image)

<table>
<thead>
<tr>
<th>SIZE OPTIMIZATION</th>
<th>With optimization</th>
<th>Without optim.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training error</td>
<td>7.728e-11</td>
<td>7.728e-11</td>
</tr>
<tr>
<td>Testing error</td>
<td>7.8093e-11</td>
<td>7.8093e-11</td>
</tr>
<tr>
<td>Time</td>
<td>41</td>
<td>37.31</td>
</tr>
<tr>
<td>Number of terms</td>
<td>2</td>
<td>2.8</td>
</tr>
</tbody>
</table>

This first example was only used to show the ability of PANN to find the model, since in fact, the May equation is an element of \( \Phi_p \).

**B. Lorenz Equation**

As a second experiment, we used Lorenz’s equations [20] with \( \sigma = 10, b = 8/3, r = 24.74 \):

\[ \begin{align*}
  x &= -\sigma x + \sigma y \\
  y &= -xz + rx - y \\
  z &= xy - bz
\end{align*} \]

We used variable \( z \) to generate a 400 values time series. In this case we used the first 250 values for the training, while the proof was made on the remaining elements. The results for the experiment follow.

<table>
<thead>
<tr>
<th>SIZE OPTIMIZATION</th>
<th>With optimization</th>
<th>Without optim.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training error</td>
<td>1.3638e-4</td>
<td>1.3597e-4</td>
</tr>
</tbody>
</table>
Testing error | 1.3654e-4 | 1.3654e-4
---|---|---
Time | 1085 | 1032
Number of terms | 56 | 133

In this case we can see an important difference between the cases. If the number of previous values or the maximum power of the polynomial is increased the difference in the number of terms with and without optimization is even larger.

VI. CONCLUDING REMARKS

The use of PANN to nonlinear time series forecasting allows us to have a better model than multilayer neural network [5], with the advantage that it is possible to have information about the behavior of the variables analyzing every term and the weights obtained by the GA and the LMS optimization. Moreover, we have shown a variation of the learning algorithm capable of reducing the number of monomials used for the solution without meaningfully increase the time needed or the mean error. This result allows different industry applications where these features are important.

REFERENCES