Time Series Forecasting using Polynomial Artificial Neural Networks

Yoltic Jassiel García Guzmán
Faculty of Engineering
Universidad La Salle
Mexico City, Mexico
yjgg@lasallistas.org.mx

***Abstract*—The Polynomial Artificial Neural network is a different approach to a design of a universal structure identificator, which uses a statistical method to adapt to a structure by means of maximum likehood techniques, giving an efficient design for data forecasting in high performance computing where the model is complex in nature.**

***Index Terms*—PANN; Time series forecasting; Neural Networks**

I. Introduction

Artificial Neural Networks, in general, are function approximations. They identify patterns in a dataset according to their relationship. An ANN is trained to correctly classify the test information, and then it is fed with new information to make an attempt of forecasting the correct answer. Polynomial neural networks, unlike ANNs, can handle the interaction between multiple layers of a neuron. It is considered a polynomial paradigm because of the multiple variables and parameters the PANN uses to transform the input into a series of weighted combinations using time-delayed dynamic periodic activation functions (Zjavka, 2011).

Polynomial approximations are used in the fields of data mining, knowledge discovery, prediction, complex systems modeling, optimization and pattern recognition. They were first introduced with the introduction of Group method of data handling (GMDH), a mathematical model of multi-parametric data sets that feature fully automatic structural and parametric optimization of models ("Group method of data handling," 2020), in the hope to capture the complexity of a process and decompose it into simple relationships described by a processing function of a neuron. Since then, other types of approximations appeared, such as NARMAX, High Order Neural Network, Non-linear interconnection, and Polynomial Neural Networks.

In section II we describe the PANN functionality, in section III we discuss the methodology used to train and test a PANN, showing the corresponding results in section IV, to later analyze the output and discuss the results in section V.

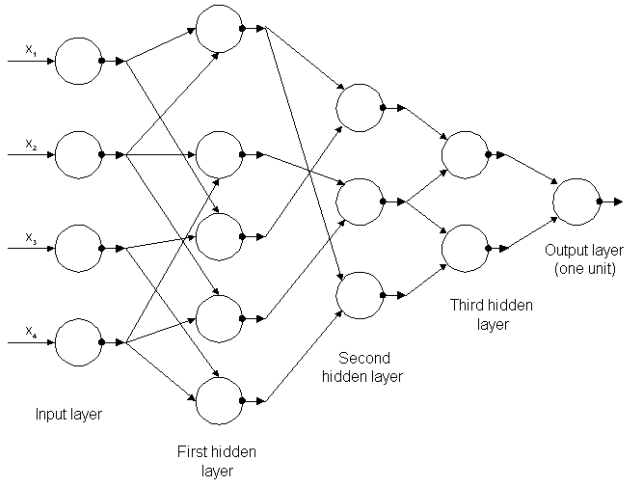

Fig. 1.1 - GMDH polynomial neural network

II. Polynomial Artificial Neural Networks

The PANN model can be described according to (Gomez-Ramirez et al., 1999) as the following equation:

$$\hat{y_k} = [\phi(x_{1,k}, x_{2,k}, ...x_{n_i,k}, x_{1,k-1}, x_{2,k-1}, ..., x_{n_i,k-n_1}, \\ ...y_{k-1}, y_{k-2}, ..., y_{k-n_2})]_{\phi_{\min}}^{\phi_{\max}}$$

- Where:
  - $\hat{y_k} \in \mathbf{R}$ is the approximation of a function, or the network output,
  - $\phi(x, y) \in \mathbf{R}$ is a non linear function,
  - $x_i \in X$ are the inputs,
  - $i = 1, \ldots, n_i; n_i =$ number of inputs,
  - $y_{k-j} \in Y$ are the previous values of the input,
  - $j = 1, \ldots, n_2, n_1$ are the number of delays in the input,
  - n_2 is the number of delays in the output,
  - X, Y are compact spaces of $\mathbf{R}$

The applied non-linear $\phi(z)$ function is described by:

$$[\phi(z)]_{\phi_{\min}}^{\phi_{\max}} = \begin{cases} \phi_{\max} & \phi(z) \geq \phi_{\max} \\ \phi(z) & \phi_{\min} < \phi(z) < \phi_{\max} \\ \phi_{\min} & \phi(z) \leq \phi_{\min} \end{cases}$$

Where $\phi_{\max}$ and $\phi_{\min}$ are maximum and minimum limits respectively.

In summary:

$$z = \{x_{1,k}, x_{2,k}, \ldots, x_{n_1,k}, \ldots, y_{k-1}, y_{k-2}, \ldots, y_{k-n_2}\} = \{z_1, z_2, z_3, \ldots, z_{n_v}\}$$

Where n_v is the number of elements on z, that is, the total amount of inputs, previous values from input and output:

$$n_v = n_i + n_1 n_i + n_2$$

Therefore, the function $\phi(z) \in \Phi_p$ can be written as:

$$\Phi_p(z_1, z_2, \ldots, z_{n_v}) = \{\phi(z) : \phi(z) = a_0(z_1, z_2, \ldots, z_{n_v}) + a_1(z_1, z_2, \ldots, z_{n_v}) + a_2(z_1, z_2, \ldots, z_{n_v}) + \ldots + a_p(z_1, z_2, \ldots, z_{n_v})\}$$

The multiple number of thresholds have a purpose on filtering out the least useful elements, to finally predict the correct output Φ (Ivakhnenko, 1971).

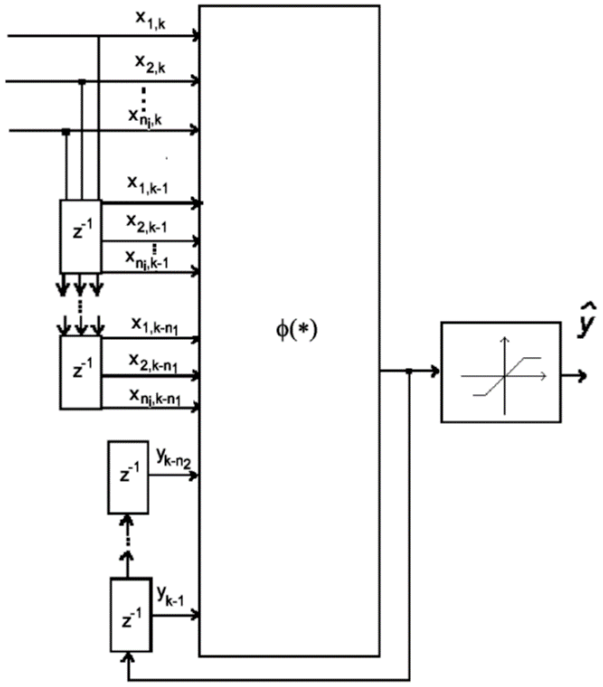

Fig. 2.1 - PANN scheme

III. Methodology

We used the GNU Octave programming language to run the simulations. This programming language is identical to MATLAB, which is mostly used in mathematics, engineering, and machine learning and neural networks. The program proportioned by (Gómez-Ramírez, 2005) was used and modified with test data proportioned by the UCI Machine Learning Repository (*UCI Machine Learning Repository*, n.d.), about the Ozone Level Detection Data from the year 2004, with samples taken every day, measuring the average ozone level. The following parameters were changed on the tests, and the results are described in the following section.

IV. Results

Test	n_{train}	n_{test}	n_{trans}	prevval	maxpow	ratio	e_{train}	e_{test}	Observations
1	150	150	50	[0, 1]	1	1	0.0035	0.0075	Initial conditions
2	150	150	50	[0, 1]	1	8	0.0041	0.0098	$e_{\text{train}} \uparrow, e_{\text{test}} \uparrow$
3	150	150	50	[0, 1]	2	1	0.0035	0.0074	$e_{\text{train}} \downarrow, e_{\text{test}} \downarrow$
4	150	150	50	[0, 1]	5	1	0.0035	0.0074	Increasing maxpow does not benefit the result
5	150	150	50	[0, 2]	5	1	0.0031	0.0099	$e_{\text{train}} \downarrow, e_{\text{test}} \uparrow$
6	150	150	50	[0, 2]	5	10	0.0032	0.0079	$e_{\text{train}} \uparrow, e_{\text{test}} \downarrow$
7	120	150	50	[0, 2]	5	10	0.0038	0.0029	$e_{\text{train}} \uparrow, e_{\text{test}} \downarrow$
8	120	120	50	[0, 2]	5	10	0.0038	0.0019	Decreasing n_{test} will decrease e_{test}since there are fewer points

Table 1 - Results of the experiment with PANN algorithm

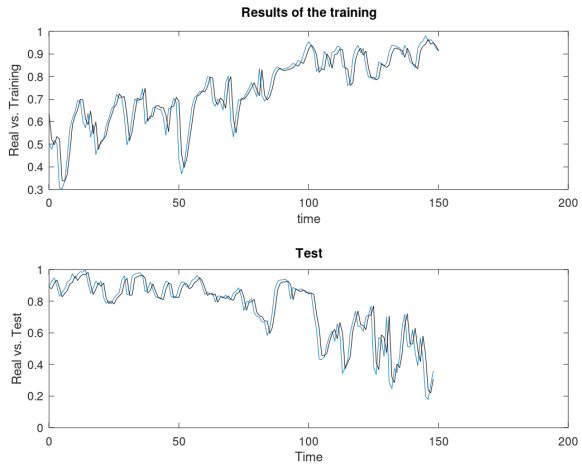

Fig. 4.1 - Test No. 1

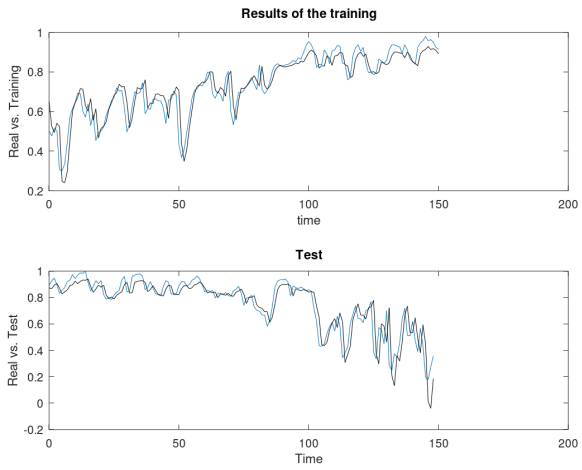

Fig. 4.2 - Test No. 2

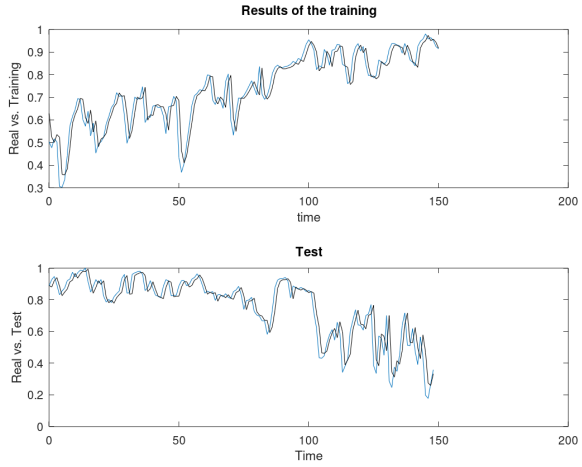

Fig. 4.3 - Test No. 3

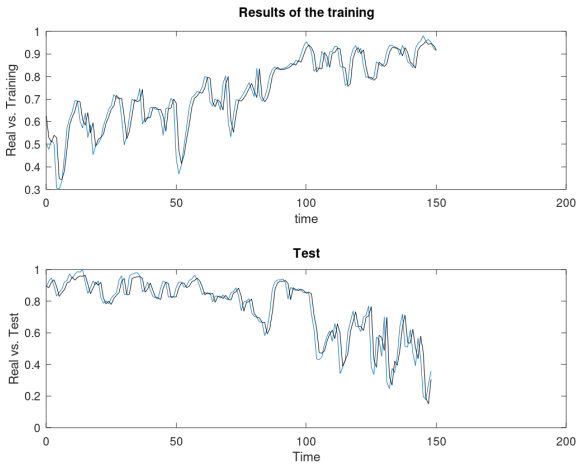

Fig. 4.4 - Test No. 4

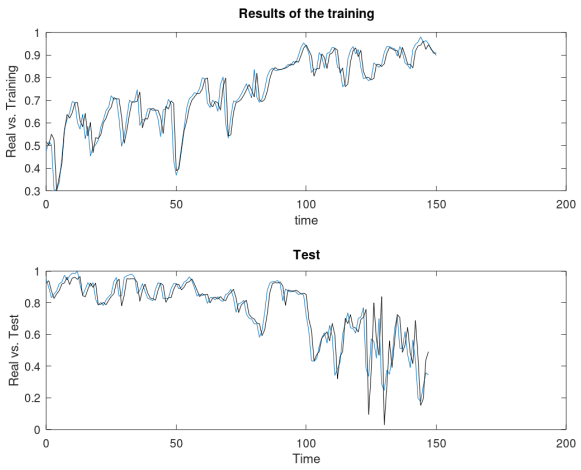

Fig. 4.5 - Test No. 5

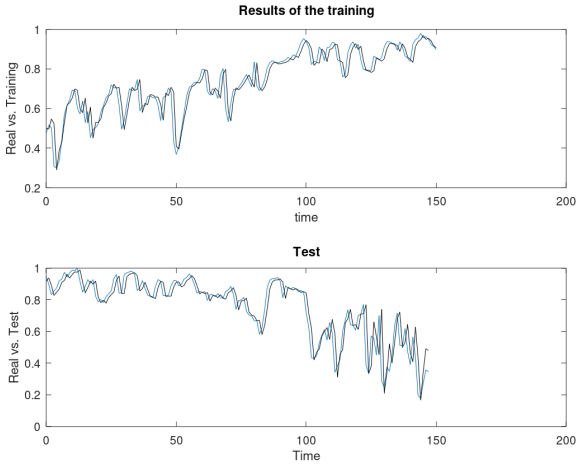

Fig. 4.6 - Test No. 6

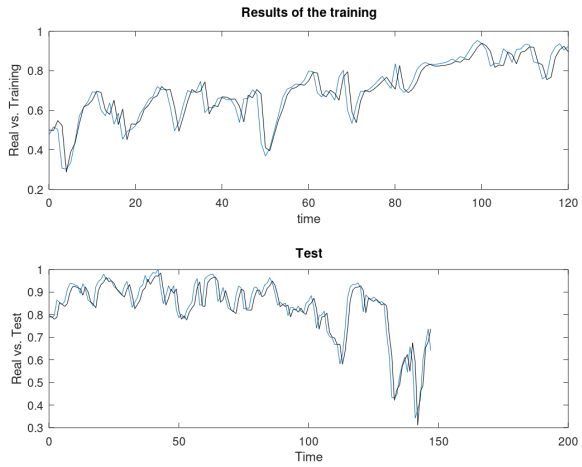

Fig. 4.7 - Test No. 7

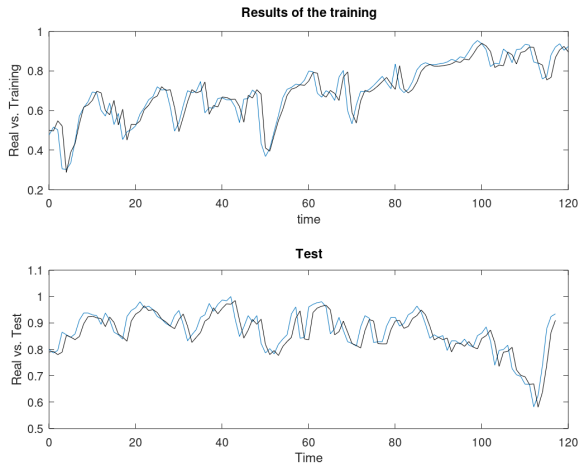

Fig. 4.8 - Test No. 8

V. Discussion

The first attempt to set the initial conditions was already approximating the original data with no significant error. However; multiple tests were run in hope to get a better result. The following times the procedure was applied with several configurations and different parameter values. The result is that PANNs do not require an extensive fine tuning to get the desired results from them. There were some cases in the experiment that made clear which parameters are directly involved with a certain type of error. For example, in test 8 was observed that decreasing the number of test points will decrease the test error, this is a consequence of lowering the thresholds by decreasing the amount of data used in the measure of error. The results of the training and testing of the PANN show us that, despite the several modifications done to reduce the error in both scenarios, the graphs present a slight displacement in time, speculatively attributed to the implementation of the algorithm. Nevertheless; it is clear from the plots and the error from both test and train, that despite this fact, the PANN is capable to make a forecast of the test values with a polynomial data fitting, and giving a correct approximation to the original data.

VI. Conclusion

PANNs can show us a great behavior with several problems of data forecasting, data fitting, pattern recognition, and general function approximations. With these networks, it is possible to create nonlinear interconnections between multiple inner layers, and the algorithm mostly handles the correct values for the optimal configuration. The result is a much more accurate representation of the problem, with fewer adjustments to do before settling on usable conditions.

References

Gomez-Ramirez, E., Poznyak, A., Gonzalez-Yunes, A., & Avila-Alvarez, M. (1999). Adaptive architecture of polynomial artificial neural network to forecast nonlinear time series. *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, 317–324. https://doi.org/10.1109/CEC.1999.781942

Gómez-Ramírez, E. (2005). *Red Neuronal Artificial Polinomial.*

Group method of data handling. (2020). *Wikipedia.*

Ivakhnenko, A. G. (1971). Polynomial Theory of Complex Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-1*(4), 364–378. https://doi.org/10.1109/TSMC.1971.4308320

UCI Machine Learning Repository: Ozone Level Detection Data Set. (n.d.). https://archive.ics.uci.edu/ml/datasets/Ozone+Level+Detection.

Zjavka, L. (2011). Differential Polynomial Neural Network. *Journal of Artificial Intelligence*, *4*(1), 89–99. https://doi.org/10.3923/jai.2011.89.99