Regression in Artificial Neural Networks

Harti Luiz Sachser Junior
Faculty of Exact Sciences and Engineering
University of Madeira
Funchal, Portugal
2069422@student.uma.pt

Felipe Clevert
Faculty of Exact Sciences and Engineering
University of Madeira
Funchal, Portugal
2016722@student.uma.pt

***Abstract*—In this document we explore three datasets found on Kaggle[1]: "Fish Market", "Monthly Milk Production" and "BitCoin Price", for each of these datasets we created 20 Artificial Neural Networks, 10 for each of 2 algorithms (Scaled Conjugate Gradient and the Resilient Backpropagation), and trained them, obtaining the values for the Performance and the Correlation Coefficient (also known as the R coefficient).**

Keywords—artificial neural network, algorithms, machine learning, training, validation, NARXNET, neurons, regression.

I. Introduction

In this project, our group was tasked with evaluating the data from the three different datasets and build several Artificial Neural Networks (ANN), with pre-determined number of neurons, using regression methods to train the model to predict a target.

Regression methods are typically used during the training phase of neural networks to optimize the model's parameters and improve its predictive performance. In the context of neural networks, regression refers to the task of predicting a continuous output variable given a set of input features. During the training phase, the neural network is trained to minimize the difference between its predicted output and the actual output, typically using an objective function such as mean squared error or mean absolute error.

The Fish Market Dataset contains 160 samples of physical data of seven species (Bream, Roach, Whitefish, Parkki, Perch, Pike, Smelt) of fish, which can be categorized based on the following attributes: *weight, vert_length, diagonal_length, cross_length, height* and *width*.

As for the monthly milk production dataset, it shows the monthly production of milk, in pounds per cow, for the years from 1962 to 1975, in 168 samples.

Lastly, we have the Bitcoin prices dataset from 2014 until 2023, showing its daily values based on the following attributes: *open, high, low, close, adj_close* and *volume*.

We need to make sure the dataset does not have inconsistencies and outliers before being able to use it on a machine learning model. Autocorrelation plots, or heatmaps, are heavily used to determine and summarize the strength of relationships between the variables or attributes [2]

II. Development

ANNs are a type of machine learning model that is loosely inspired by the structure and function of biological neural networks, such as the human brain. ANNs consist of interconnected nodes (called neurons) that perform simple computations on the input data. The neurons are organized into layers, with the input layer receiving the raw input data, one or more hidden layers processing the data through a series of non-linear transformations, and the output layer producing the final output of the network.

The weights of the connections between neurons are adjusted during training using an optimization algorithm, such as backpropagation, to minimize a specified loss function that measures the difference between the predicted output of the network and the true output. This process allows the network to learn the underlying patterns and relationships in the input data and produce accurate predictions on new, unseen data.

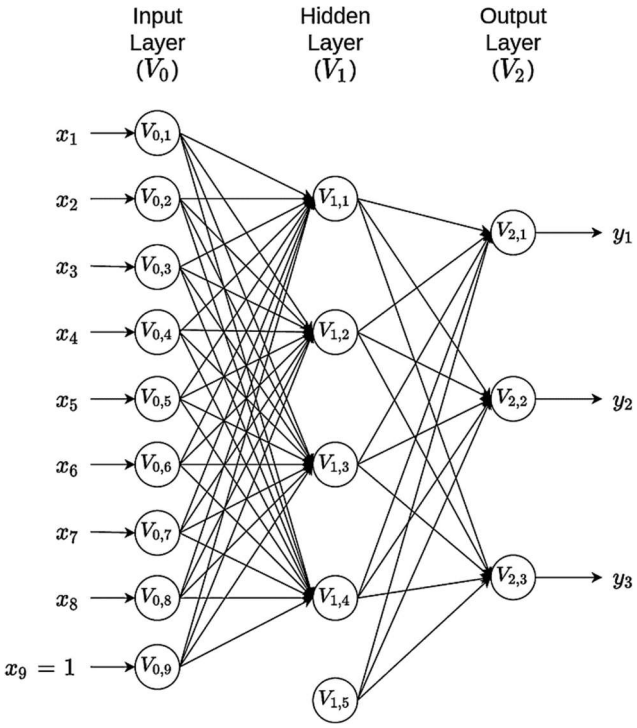

Figure 1- Example of ANN

$x_1, \ldots, x_p$ represents the information (input) that the neuron receives from the external sensory system or from other neurons with which it has a connection. $w = (w_1, \ldots, w_p)$ is the vector of synaptic weights that modifies the received information emulating the synapse between the biological neurons. These can be interpreted as gains that can attenuate or amplify the values that they wish to propagate toward the neuron.

Fig. (1) shows that in the learning process of an artificial neural network are involved the interaction of layers, input data, loss function which defines the feedback signal used for learning, and the optimizer which determines how the learning proceeds and uses the loss value to update the network's weights. Initially, the weights of the network are assigned small random values, but when this provides an output far from the ideal values, it also implies a high loss

score. But at each iteration of the network process, the weights are adjusted a little to reduce the difference between the observed and predicted values and, of course, to decrease the loss score. This is the basic step of the training process of statistical machine learning models in general, and when this process is repeated a sufficient number of times (on the order of thousands of iterations sometimes), it yields weight values that minimize the loss function.

Layers and neurons are separated as follows:

- Input layer - It is the set of neurons $V_{01}…V_{0n}$ that directly receives the information coming from the external sources of the network. Therefore, the number of neurons in an input layer is most of the time the same as the number of the input variables provided to the network;
- Hidden layer - Consist of a set of internal neurons of the network that do not have direct contact with the outside. The number of hidden layers can be 0, 1, or more. The learned information extracted from the training data is stored and captured by the weight values of the connections between the layers of the artificial neural network. Also, it is important to point out that hidden layers are key components for capturing complex nonlinear behaviors of data more efficiently;
- Output Layer - It is a set of neurons that transfers the information that the network has processed to the outside. the output neurons correspond to the output variables y_1,….. y_n. This means that the output layer gives the answer or prediction of the artificial neural network model based on the input from the input layer.

III. Algorithms

The ANNs perform based on a modelled function, more specifically, a *cost function;* a cost or what is sometimes called a loss function measures how wrong a model is. The partial derivatives of the cost function influence the ultimate model's weights and biases selected. The error gap between the predicted value of a neural network and the actual value of a data sample is facilitated by the cost function. Examples of cost functions: Mean Squared Error, Categorical Cross-Entropy and Logarithmic Loss.

The image below illustrates a cost function that holds values of the possible parameters of weight and biases.

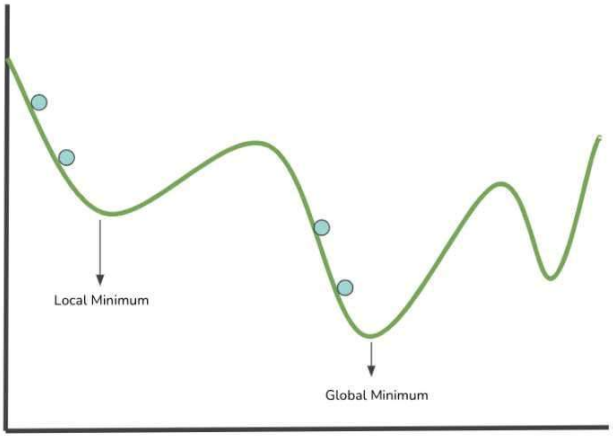

Figure 2- Example of learning algorithm

The blue points in the image above represent a step (evaluation of parameters values into the cost function) in the search for a local minimum. The lowest point of a modelled cost function corresponds to the position of weights values that results in the lowest value of the cost function. The smaller the cost function is, the better the neural network performs.

A. *NARXNET (Nonlinear Autoregressive with eXogenous inputs*

NARX is a type of ANN architecture used for time-series prediction NARX network is a type of recurrent neural network (RNN) that has feedback connections from the output layer to the hidden layer, allowing the network to use its own predictions as inputs for future predictions.[3]

The NARX network has a similar architecture to a traditional feedforward neural network, but with the addition of feedback connections. It takes past input and output data as well as current exogenous input data and uses them to predict the future output. The network is trained using a supervised learning algorithm, such as backpropagation, to minimize the difference between the predicted and actual outputs.

NARX networks are commonly used for time-series prediction problems where the input and output variables exhibit some nonlinear dependencies. They have been successfully applied to various applications such as financial forecasting, stock market prediction, weather forecasting, and many others.

For the last two datasets we ae going to use 2nd order model, making an ANN with higher capacity and somehow dynamic. This means that as inputs, the model used the current and past values of the same series.

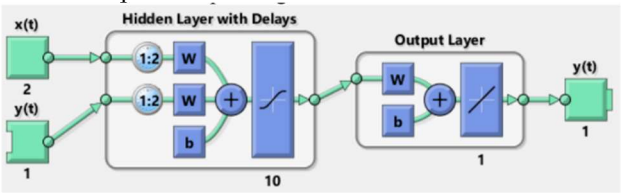

Figure 3- NARXNET

B. *Resilient Backpropagation*

The training process of ANN, which consists of adjusting connection weights, requires a lot of computational resources. Some advantages of this algorithm are (a) it is able to make predictions of categorical or continuous outcomes, (b) it does a better job in capturing complex patterns than nearly any other algorithm, and (c) few assumptions about the underlying relationships of the data are made. However, this algorithm is not without weaknesses, some of which are (a) it is very slow to train and (b) requires a lot of computational resources. The algorithm basically uses the partial derivatives of the loss function with respect to the weights, which represent the rate of change of the loss function (the slope of the loss function. The goal is to apply this method until the score of the loss function no longer decreases, finding possible local and/or global minimum points (see Fig. 2).

The resilient backpropagation is an optimization of the standard backpropagation, designed to overcome some of the limitations such as slow convergence and sensitivity to the choice of learning rate. The key idea behind it is to adjust the weights of the neural network based on the sign of the gradient, rather than the magnitude. Specifically, the

algorithm maintains a separate update value for each weight, which is initialized to a small positive value. During training, the sign of the gradient is used to determine whether the weight should be increased or decreased, and the update value is adjusted accordingly. If the gradient remains the same sign as in the previous iteration, the update value is increased, which leads to a larger weight update. Conversely, if the gradient changes sign, the update value is decreased, which results in a smaller weight update.[4]

C. *Scaled Conjugate Gradient*

The scaled conjugate gradient (SCG) algorithm is based on conjugate directions, but this algorithm does not perform a line search at each iteration unlike other conjugate gradient algorithms which require a line search at each iteration, making the system computationally expensive. SCG was designed to avoid the time-consuming line search.

It can train any network as long as its weight, net input, and transfer functions have derivative functions. In SCG algorithm, the step size is a function of quadratic approximation of the error function which makes it more robust and independent of user defined parameters. The step size is estimated using different approaches.

IV. Neural Networks

For this project we had to create 10 ANNs for each algorithm and dataset, totaling 60 ANNs. All of them were created with random weights between a determined range and a random number of neurons for the hidden layer. In order to implement this onto code, we used two "for loops", one to change between the 2 selected algorithms (Resilient Backpropagation and Scaled Conjugate Gradient) and the other to create the 10 ANNs for each algorithm.

The random initial weights were given a range as follows:

$$\left[-\frac{2.4}{I}, \frac{2.4}{I}\right] \qquad (1)$$

Where I is the number of inputs of the ANN model for each dataset.

Regarding the number of neurons of the hidden layer, we used a random number to choose from an array N_n of values:

$$N_n = [4, 7, 10, 12, 15, 20] \qquad (2)$$

For each element of the array above, it will be assigned an index number from a vector [1,…,6]

A. *Fish Market – Fitting ANN*

This dataset contains various information about fish size dimensions and weight for seven different species.

The first step was to look for possible missing values and any inconsistencies. After that, we categorized the *Species* column mapping each string to a number [0-6], as the model cannot take strings as input.

We now have all the columns in the dataset as numerical values and *Weight* is going to be our target variable. To get a more balanced dataset, we normalized the target variable using the Min-Max method.

Our model probably does not need all the features and we can confirm that using a Correlation matrix to check the relationship between them as follows:

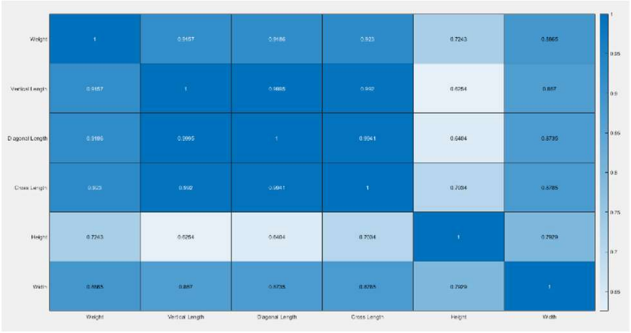

Figure 4- Features Correlation Matrix

Checking Fig. 4 above, we then decided not to use the *Height, Width and Cross Length* attributes. The decision is based on the first two having a weak correlation to our predictable variable (*Weight*), and the last one because albeit having a stronger correlation comparing to V*ertical Length and Diagonal Length,* the correlation between themselves is really strong and it would not affect the results if we excluded one of them.

Having decided which variables the model is going to use, now it was time to find some outliers and replace them using the Linear Interpolation technique. We coded it in Matlab, using the function '*filloutliers'* together with the argument *'linear'*.

Using the data obtained by training the 20 ANNs, we developed the following tables containing information of the network regarding number of neurons used, performance and the R Coefficient, which evaluates how good the model can predict the target variable.

For the first algorithm (SGC) the table below is obtained:

Table 1- SGC for First Dataset

Network	Neurons	MSE	R Coef.
1	4	0,0084792	0,9451900
2	12	0,0057049	0,9641770
3	20	0,0140511	0,9164502
4	20	0,0045510	0,9714296
5	15	0,0047312	0,9698015
6	7	0,0091776	0,9407858
7	7	0,0087482	0,9446955
8	4	0,0072594	0,9532477
9	15	0,0056373	0,9639139
10	7	0,0109936	0,9353817

Analyzing the table above we want to distinguish the best and worst parameters, as follows:

Table 2- Worst and Best SCG - First Dataset

	Network	Neurons	MSE	R Coef.
Worst	3	20	0,0140511	0,9164502
Best	4	20	0,0045510	0,9714296

The same can be done for the second algorithm (RBP), as follows:

Table 3- RBP for First Dataset

Network	Neurons	MSE	R Coef.
1	4	0,0087536	0,9433846
2	15	0,0097096	0,9368991
3	15	0,0088532	0,9428531
4	20	0,0087088	0,9440729
5	4	0,0088432	0,9426895
6	15	0,0086139	0,9463054
7	12	0,0084877	0,9452966
8	10	0,0141691	0,9078784
9	12	0,0093951	0,9391377
10	15	0,0087859	0,9432140

Analyzing the table above we want to distinguish the best and worst parameters, as follows:

Table 4- Worst and Best RBP - First Dataset

	Network	Neurons	MSE	R Coef.
Worst	8	10	0,0141691	0,9078784
Best	6	15	0,0086139	0,9463054

B. Monthly Milk Production – Time Series ANN

For this dataset, namely Milk Production, we analyzed the monthly production of milk in pounds per cow, over a period from 1961 to 1975. Our target variable would be production and as such, the data was normalized using the Min-Max method. Other than that, the column related to the time period had to be converted into *'year-month'* as a numerical value.

This dataset is a time series and a NARXNET network, as explained previously, will be used with pre-defined parameters:

$$inputDelays = 2 \qquad (3)$$

$$feedbackDelays = 2 \qquad (4)$$

The parameters above in Eqs. (3) and (4) refer to the 2nd order model. The rest of the parameters were set to work as the previous dataset.

The initial plot shows us how the dataset can be represented already normalized, as follows:

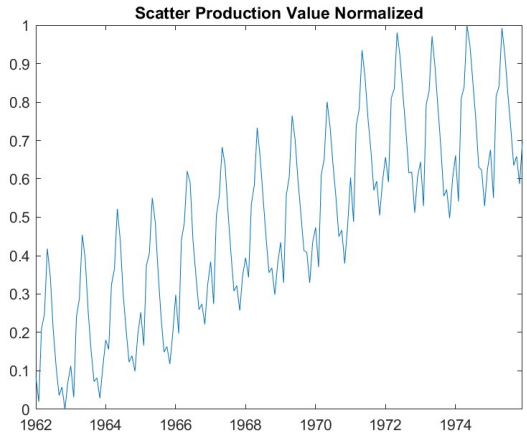

Figure 5- Scatter plot of Milk Production

Using the data obtained by training the 20 ANNs, we developed the following tables containing information of the network regarding number of neurons used, performance and the R Coefficient, which evaluates how good the model can predict the target variable.

For the first algorithm (SGC) the table below is obtained:

Table 5- SGC for Second Dataset

Network	Neurons	MSE	R Coef.
1	10	3064,780244	0,8358322667
2	4	2754,908281	0,8530939888
3	20	2551,253805	0,8659824267
4	4	2749,938081	0,8532869533
5	4	2980,318885	0,8400143371
6	12	3039,49899	0,8381769119
7	7	2466,859895	0,8703955559
8	15	3045,732096	0,8370058279
9	15	2778,785199	0,8537767092
10	15	3345,452453	0,8202751169

Analyzing the table above we want to distinguish the best and worst parameters, as follows:

Table 6- Worst and Best SCG - Second Dataset

	Network	Neurons	MSE	R Coef.
Worst	10	15	3345,452453	0,8202751169
Best	7	7	2466,859895	0,8703955559

The same can be done for the second algorithm (RBP), as follows:

Table 7- RBP for Second Dataset

Network	Neurons	MSE	R Coef.
1	7	2989,128845	0,8419910079
2	4	2671,694218	0,8582137231
3	15	2682,785013	0,8574068310
4	7	3185,459437	0,8383623053
5	10	3000,234762	0,8395590873
6	15	2956,749218	0,8423973847
7	4	2847,979536	0,8478029504
8	10	2639,113724	0,8610842041
9	4	2740,091632	0,8559984367
10	7	2667,041391	0,8582912036

Analyzing the table above we want to distinguish the best and worst parameters, as follows:

Table 8- Worst and Best RBP - Second Dataset

	Network	Neurons	MSE	R Coef.
Worst	4	7	3185,459437	0,8383623053
Best	8	10	2639,113724	0,8610842041

C. Bitcoin Price – Time Series ANN

This dataset consists of many attributes regarding the price of Bitcoin over time from its creation until 2023. In this case we are going to be using the *'Close Price'* as our target variable. Like in the previous dataset, the data was normalized using the Min-Max method, the column related to the time period had to be converted into *'year-month-day'* as a numerical value and we used a NARX network with 2nd order for a time series. See Eqs. (3) and (4).

The initial plot shows us how the dataset can be represented already normalized, as follows:

Figure 6- Bitcoin Prices

Using the data obtained by training the 20 ANNs, we developed the following tables containing information of the network regarding number of neurons used, performance and the R Coefficient, which evaluates how good the model can predict the target variable.

For the first algorithm (SGC) the table below is obtained:

Table 9- SGC for Third Dataset

Network	Neurons	MSE	R Coef.
1	15	1166325,1963	0,9977319047
2	7	1194118,4075	0,9976842710
3	12	1202618,5985	0,9976615977
4	15	1149432,5477	0,9977697038
5	12	1159883,4700	0,9977444652
6	12	1161033,7979	0,997744699
7	10	1184734,1119	0,9976976030
8	20	1164463,2598	0,9977356284
9	15	1197024,9296	0,9976747884
10	20	1144274,9584	0,9977755398

Analyzing the table above we want to distinguish the best and worst parameters, as follows:

Table 10- Worst and Best SCG - Third Dataset

	Network	Neurons	MSE	R Coef.
Worst	3	12	1202618,5985	0,9976615977
Best	10	20	1144274,9584	0,9977755398

The same can be done for the second algorithm (RBP), as follows:

Table 11- RBP for Third Dataset

Network	Neurons	MSE	R Coef.
1	7	1203629,0092	0,9976595728
2	7	1200383,9602	0,9976669965
3	12	1176720,5436	0,9977118705
4	10	1180297,4599	0,9977050274
5	12	1182383,7708	0,9977059696
6	7	1185399,8808	0,9976949167
7	15	1186919,2758	0,9976919892
8	20	1178778,7396	0,9977080200
9	10	1186659,6057	0,9976940719
10	12	1171106,7664	0,9977235900

Analyzing the table above we want to distinguish the best and worst parameters, as follows:

Table 12- Worst and Best RBP Third Dataset

	Network	Neurons	MSE	R Coef.
Worst	1	7	1203629,0092	0,9976595728
Best	10	12	1171106,7664	0,9977235900

V. Data Analysis

A. First Dataset

The following plots refer to the best (Fig. 7-8) and worst (Fig. 9-10) networks for Algorithm SCG.

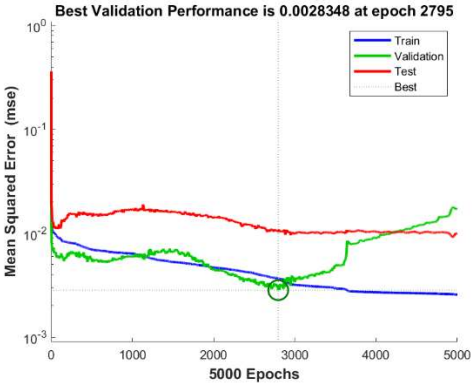

Figure 7-Best SCG

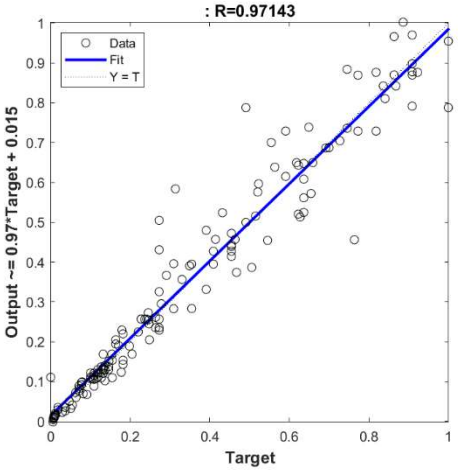

Figure 8-Best Regression SCG

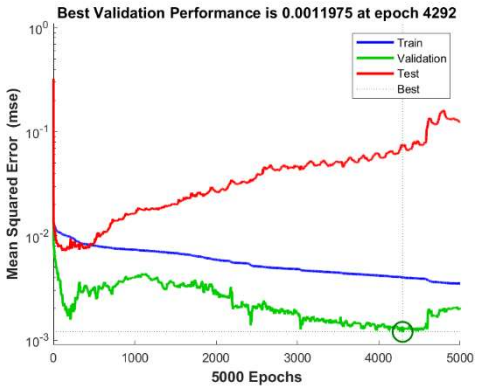

Figure 9- Worst SCG

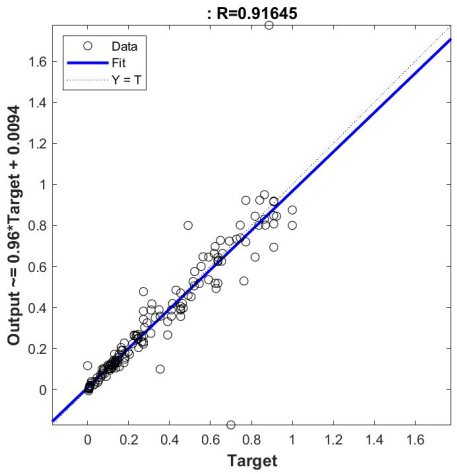

Figure 10 Worst Regression SCG

The following plots refer to the best (Fig. 11-12) and worst (Fig. 13-14) networks for Algorithm RBP.

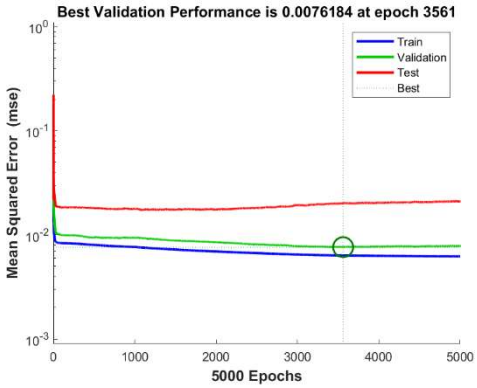

Figure 11- Best RBP

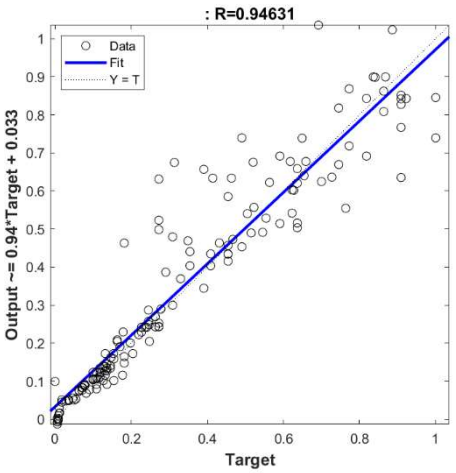

Figure 12- Best Regression RBP

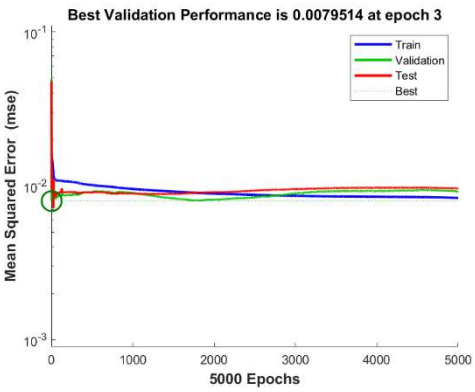

Figure 13 - Worst RBP

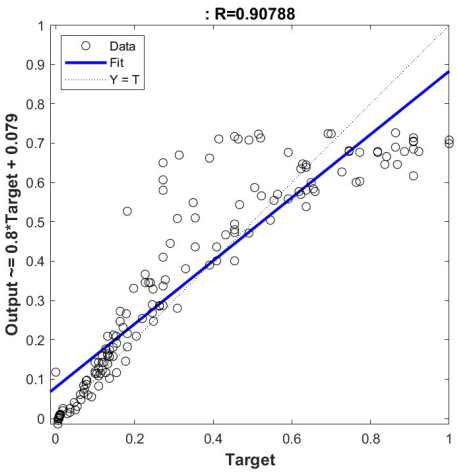

Figure 14- Worst Regression RBP

B. Second Dataset

The following plots refer to the best (Fig. 15-17) and worst (Fig. 18-20) networks for Algorithm SCG.

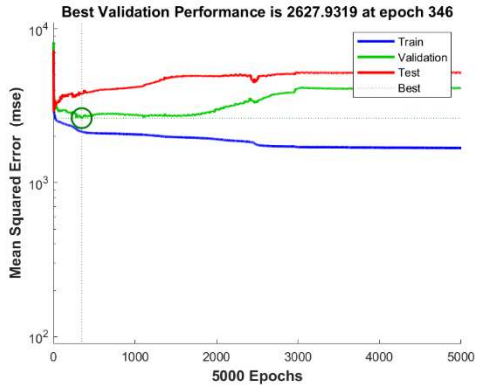

Figure 15- -Best SCG

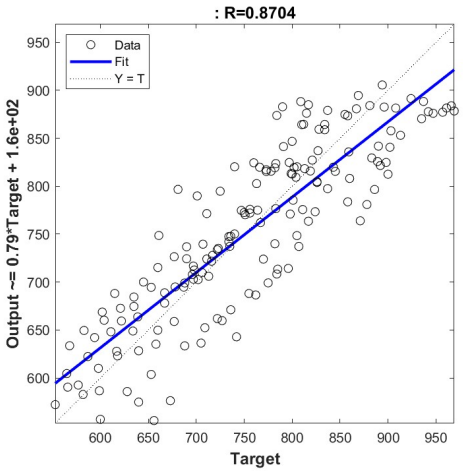

Figure 16- Best Regression SCG

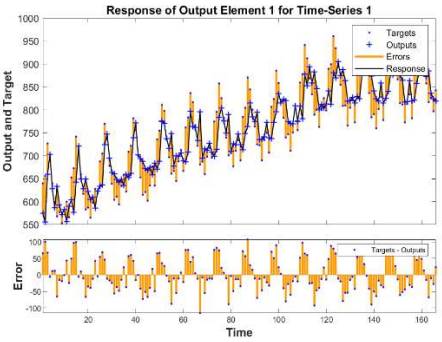

Figure 17- Best Response SCG

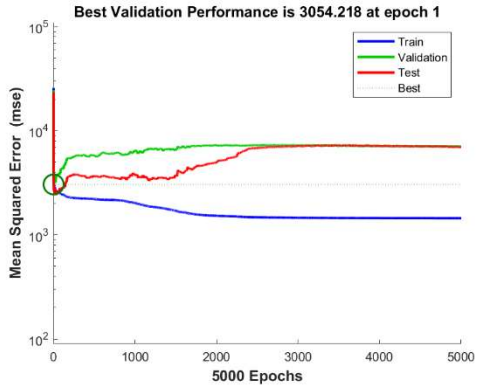

Figure 18- Worst of SCG

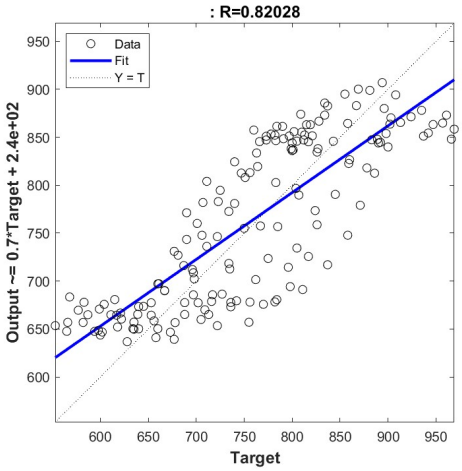

Figure 19- Worst Regression SCG

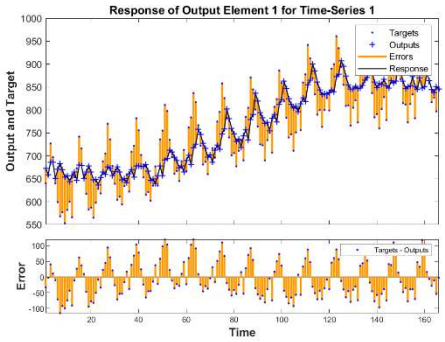

Figure 20- Worst Response SCG

The following plots refer to the best (Fig. 21-23) and worst (Fig. 24-26) networks for Algorithm RBP

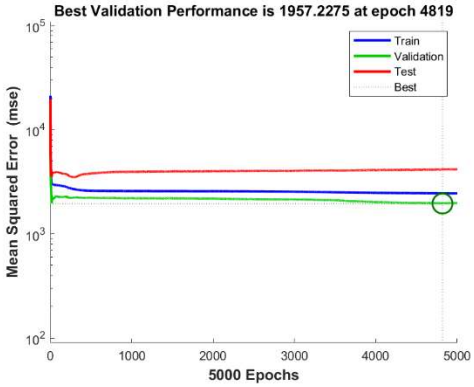

Figure 21- Best RBP

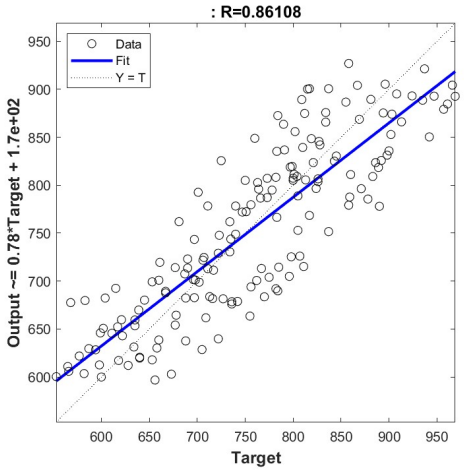

Figure 22- Best Regression SCG

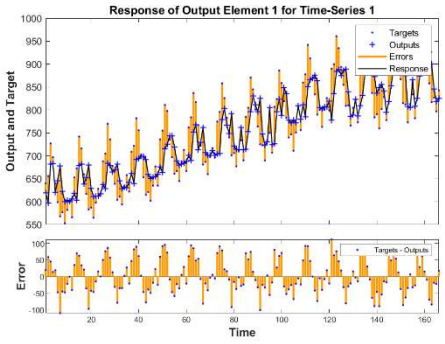

Figure 23- Best Response SCG

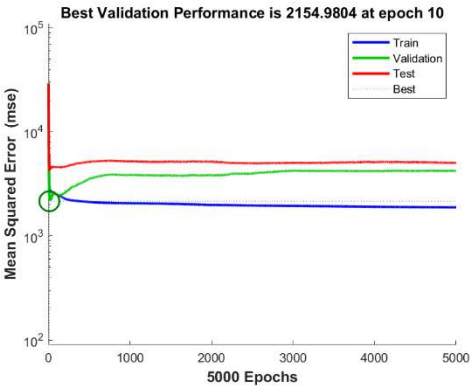

Figure 24- Worst RBP

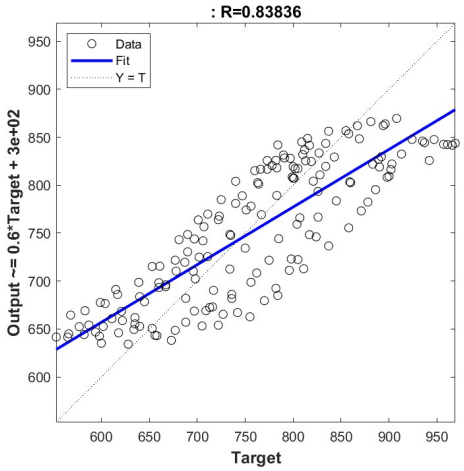

Figure 25- Worst Regression RBP

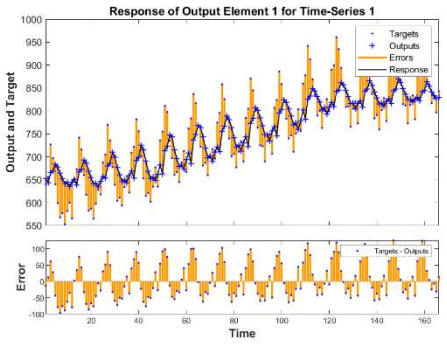

Figure 26- Worst Response RBP

C. *Third Dataset*

The following plots refer to the best (Fig. 27-29) and worst (Fig. 30-32) networks for Algorithm SCG

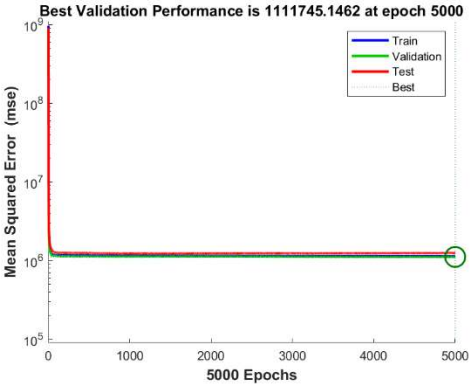

Figure 27- Best SCG

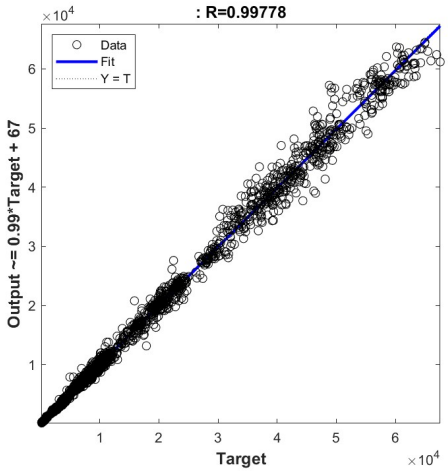

Figure 28- Best Regression SCG

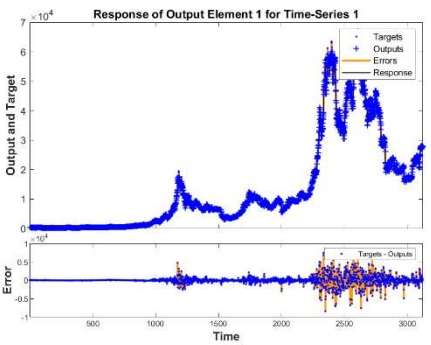

Figure 29- Best Response SCG

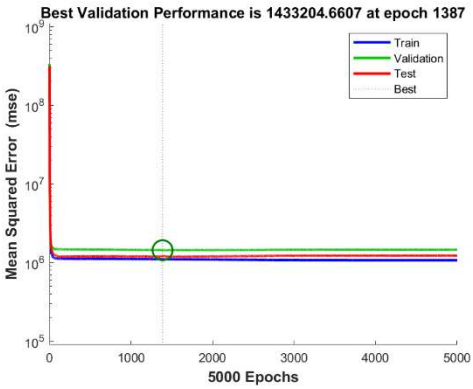

Figure 30- Worst SCG

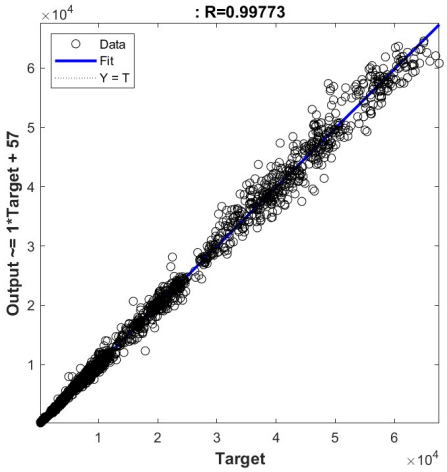

Figure 31- Worst Regression SCG

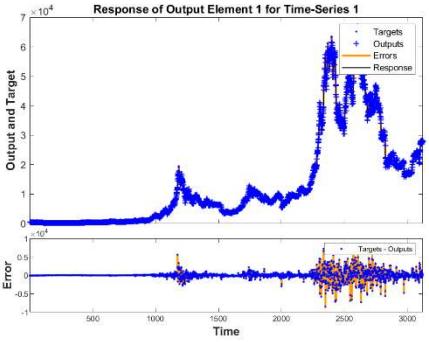

Figure 32- Worst Response SCG

The following plots refer to the best (Fig. 33-35) and worst (Fig. 36-38) networks for Algorithm RBP.

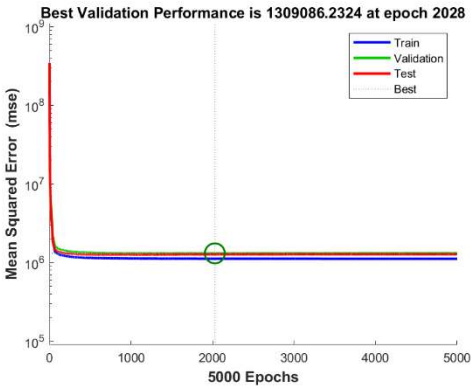

Figure 33- Best RBP

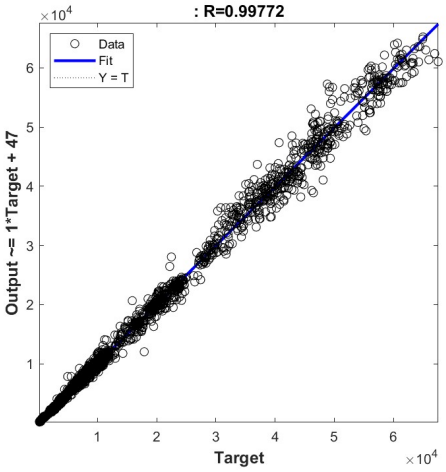

Figure 34- Best Regression RBP

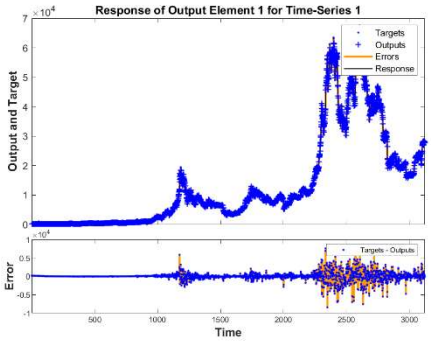

Figure 35- Best Response RBP

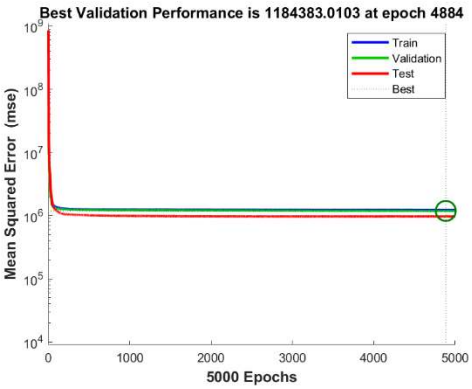

Figure 36- Worst RBP

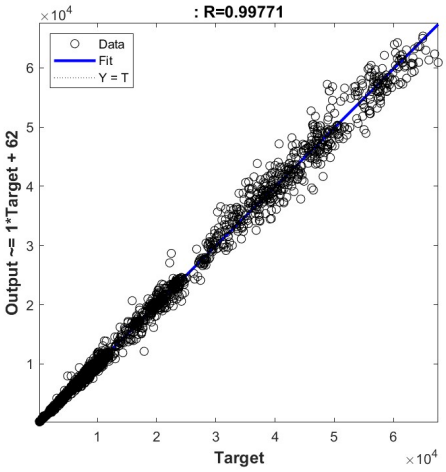

Figure 37- Worst Regression RBP

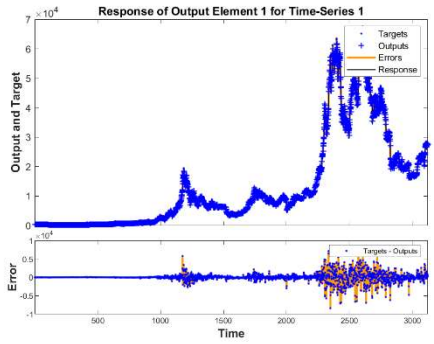

Figure 38- Worst Response RBP

CONCLUSIONS

This assignment was devised to give the students a basic understanding of machine learning, covering a wide of concepts from statistical techniques, training algorithms, as well as different types of output graph plots and its importance in machine learning model, regarding data prediction.

Both the algorithms provide fast initial convergence, however, the calculations and training methods are different in both. Results show that they are very similar when comparing the R coefficients.

Though the computational efforts are more in SCG algorithm, it achieves faster learning as against RBP algorithm due to the absence of line search optimization.

REFERENCES

[1] https://www.kaggle.com

[2] https://medium.com/analytics-vidhya/cleaning-and-understanding-multivariate-time-series-data-6554eefbda9c/

[3] https://www.mathworks.com/help/deeplearning/ug/design-time-series-narx-feedback-neural-networks.html

[4] Saputra, W. et al 2017. Journal of Physics, Conference Series 930 012035. Analysis Resilient Algorithm on ANN backpropagation