Functional Approximation using Artificial Neural Network

Aneesah Abdul Kadhar
MACS
Heriot Watt University

Dilani Maheswaran
MACS
Heriot Watt University

***Abstract*—This paper examines the use of a metaheuristic search strategy- Particle Swarm Optimization for the training of an Artificial Neural Network for functional approximation.**

***Keywords*—Artificial Neural Network, Particle Swarm Optimisation, functional approximations**

I. Introduction

Function approximation is the technique of mapping an underlying unknown function. The input is mapped to an output from a collection of input-output dataset. Instead of denoting the function explicitly, only the sets of input-output data in the form of (x,f(x)) are available. [9]

Neural Networks are a case of a supervised learning algorithm and seek to approximate the function represented in the data.

Particle Swarm Optimization is employed to train the neural networks so that the weights and biases can be adjusted.

According to Kolmogorov a multilayer perceptron can approximate any function even when the number of hidden neurons is unknown.

II. PROGRAM DEVELOPMENT RATIONALE

A. *Artificial Neural Network*

An artificial neural network is a collection of neurons organized into input, output, and hidden layers. Each neuron in these layers are connected by a set of synaptic weights.

The ANN during its training process continuously changes its weights until the goal error value is accomplished or maximum iteration is reached. Metaheuristic method such as Particle Swarm Optimization is employed for training the neural network in this study.

ANNs have been employed in various applications, especially in issues related to function approximation, due to its ability in obtaining the pattern within input-output data without the need for predetermined models.[9]

According to Cybenko [1] and Hornik [2], there exists a neural network with 1 hidden layer that is capable of approximating continuous function f with desired accuracy. Among all the models of ANNs, the multilayer perceptron (MLP) is frequently used.

Algorithm 1: Pseudocode for ANN

```
1. BEGIN
2. Parameter setting and initialization of the
   network. (i.e., weights and biases)
3. For each training data
   Forward propagation
       Find sum of dot product - equation (1)
       evaluate activation function
```

B. *Particle Swarm Optimisation*

PSO proposed by Kennedy and Eberhart [3] is a metaheuristic algorithm based on the concept of swarm intelligence (fish schooling or bird-flocking), capable of solving complex optimization problems. PSO uses a simple mechanism that simulates swarm behavior in birds flocking to search for a globally optimal solution.

In PSO, the system has a population of random solutions called particles. Each particle is given a random velocity and moves around the wide-area of search-space as per the objective function. Each particle corresponds to an encoding of neural network synapse weight, each axis for a particle position is a set of network weights and biases [4][11]. Particles have memory and each particle keeps track of the former best position and corresponding fitness. The previous best value is denoted as p_{best}. The value g_{best} denotes the best value of all particles p_{best} in the swarm. The basic concept of PSO technique lies in accelerating each particle towards its p_{best} and g_{best} locations at each iteration [4]. To train the neural network mean square error was used as the fitness function [11].

There are many ways to train the neural network for the purpose of functional approximation, this paper details the use of PSO as a training algorithm

Figure 1 describes the concept of PSO. P^k denotes the current position, P^{k+1} denotes the new position V_{ini} is initial velocity and V_{mod} is the modified velocity. V_{pbest} is velocity considering p_{best}, and V_{gbest} is velocity considering g_{best} [8].

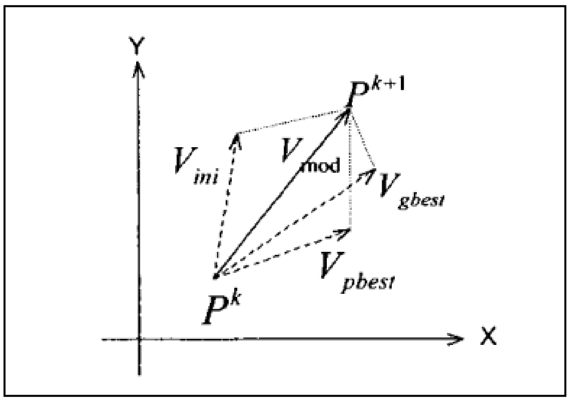


Fig 1: Concept of changing a particle's position in PSO. Source [9]

Particle swarm optimization consists of three parameters- inertial weight (w), a cognitive coefficient (c1) and social coefficient (c2).
Inertial weight estimates how immune to change a particle has in velocity i.e., a higher inertial *weight* will take the particles existing velocity into consideration, whereas a lower inertial weight will cause the particles' new velocity to contribute more to the particles' new position. C1 and C2 are positive acceleration coefficients and are mainly employed to control the balance of an individual's self-learning versus learning from the entire PSO population. [8]

Algorithm 2: Pseudocode for PSO

```
1.  BEGIN
2.  Parameter setting and initialization of swarm.
3.  Evaluate fitness and locate the leader (i.e., initialize $p_{best}$ and $g_{best}$).
4.  I=0 /* I = Iteration Count */
5.  WHILE (the stopping criterion is not met, say, I < Imax)
6.  DO
7.  FOR each particle
8.  Update position & velocity (flight) as per equations (4) & (5)
9.  Evaluate fitness
10. Update $p_{best}$
11. END FOR
12. Update leader (i.e., $g_{best}$)
13. I++
14. END WHILE
15. END
```

III. METHODOLOGY

A. Multi-layer Perceptron

A Multilayer Perceptron consists of 3 or more layers: an input layer, an output layer, and hidden layers. All the nodes from a layer are connected to the next layer (except the output layer). All nodes in the layers (except the input layer) are equipped with a non-linear function activation function.

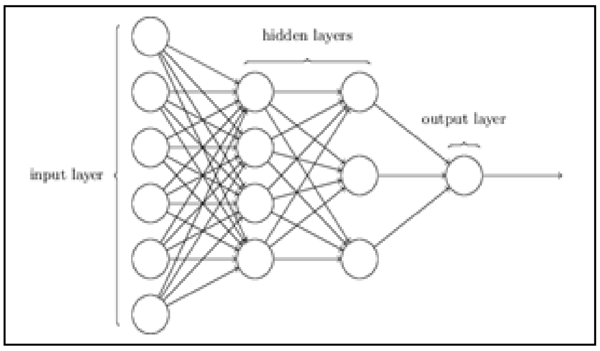


Fig 2: A Multi – layer perceptron

B. ANN Influential Factors

- The percentage of training data
- The number of neurons and the layers
- The activation functions

C. Feed-Forward Method

The weights determine the strength of the connections from each input. The output in the feed forward path is computed by the sum of the weighted inputs with a bias term added to it, which is then passed through an activation function f. The activation function is typically nonlinear, e.g. a sigmoidal function.

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$ - eq (1)

The number of Hidden Layers and number of neurons therein are passed as inputs from the user. The input and output layers are selected by the program dependent upon the data. The Dimensionality for the given problem is given by equation (2). Dimensionality for a given problem consisting of an input, hidden, output layer is given by

$$Dim = HL * (IL + 1) + (OL * (HL + 1))$$ – eq(2)

Where,

HL – Hidden Layer

IL – Input Layer

OL – Output Layer

Error function for a feedforward neural network is given by

$$E(f) = \frac{1}{2}\sum_{i=1}^{N}(d_i - y_i)^2 = \frac{1}{2}\sum_{i=1}^{N}[(d_i - f(x_i))]^2 \quad \text{eq (3)}$$

[11]

Where y_i is the actual response and d_i is the target response

D. Activation function

One of the important components of a neuron in artificial neural networks (ANN) is the activation function which alters the output of a neuron that in turn affects the performance of the neural network. Activation functions determine the output of a deep learning model, its accuracy, and the computational efficiency of training a model—which can make or break a large-scale neural network.

E. Particle Swarm Optimization

Algorithm 2 outlines the implementation of basic PSO incorporated in this study. PSO is an iterative algorithm where particles continuously explore the search space. Each particle has a position in the search space and stores personal best (p_{best}), which is the best search point the particle has experienced in its search trajectory. The particles also have a velocity in the search space and the new position is calculated by adding the velocity to the particle's current search point at every iteration. For each dimension d of each particle i, the velocity is updated at each iteration using the equation:

$$v_{id}(t+1) = wv_{id}(t) + c_1 rand_1()(p_{id}(t) - x_{id}(t)) + c_2 rand_2()(p_{gd}(t) - x_{id}(t))$$

-eq (4)

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$$

- eq (5)

where x is the particle's current position, p_{best} is the particle's personal best, r1 and r2 are uniformly distributed random variables, and ω is inertial weight, c1 and c2 are acceleration coefficients.

Each particle is assigned to a group of other particles known as informants. g_{best} is the best personal best amongst this group of particles. Informants are randomly assigned at the start of the run, implying that particles tend to have overlapping groups of informants.

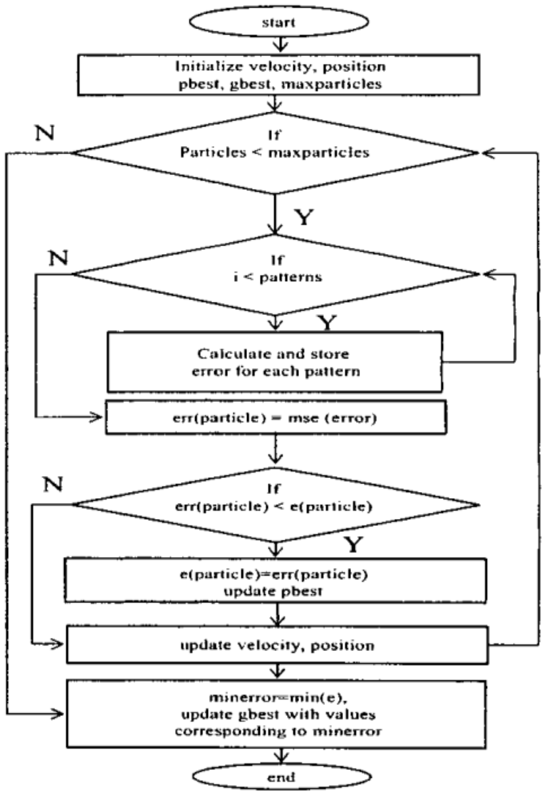

start
Initialize velocity, position pbest, gbest, maxparticles
N
If Particles < maxparticles
Y
N
If i < patterns
Y
Calculate and store error for each pattern
err(particle) = mse (error)
N
If err(particle) < e(particle)
Y
e(particle)=err(particle) update pbest
update velocity, position
minerror=min(e), update gbest with values corresponding to minerror
end

Fig 3: Training of neural network using PSO source [8]

Training Data

The data are divided into two subsets: training and testing set. The training set is used for computing the mean squared error, which is the fitness of PSO and updating the network weights and biases. After the specified number of iterations, the training is stopped, and the weights and biases of the epoch, with minimum validation errors are returned as the final ANN structure [11].

In this study all input data uses the same holdout ratio of 70% training data and 30% testing data. This provides a big sample size for the neural network to train upon and tests how well the network can functionally approximate the data it has not yet seen.

An assumption about the optimized network particle weights are made, the weights are comparatively small in magnitude and so a boundary is established. This is done so that the particles remain within the bounded region since there is no better solution outside of this region. The weights and biases are assigned randomly at the beginning [11]. Suppose a particle's position is outside of the bounded region, its fitness is not updated. This is done so that the particles remain within the bounded region, hence there is no better solution outside of the region [11].

The termination condition used for all datasets will be reaching a mean squared error equal to or less than 0.010 or until all the specified number of iterations are completed.

IV.RESULTS

Several experiments were conducted by varying the hyper parameters of both PSO and ANN to evaluate the performance of the proposed solution and to identify the influence of each hyper parameter on the accuracy and performance.

Training dataset at 50% provided evidence of underfitting. Underfitting occurs when the model is incapable of capturing the variability of the data [6]. When too many neurons were incorporated in the hidden layer, computational time (training time) increased and the network over fits the data. When over fitting occurs, the network will begin to model random noise in the data.

PSO is a stochastic algorithm and so to achieve optimum values the experiment was conducted ten times. Mean of MSE over 10 iterations was calculated. PSO parameters are recorded only when the targeted accuracy is achieved 6 out of 10 times. The selection of PSO parameters plays a vital role in the optimization of the fitness function. The optimal values for the parameters are learned by trial-error experimentations. The intertial weight w whose range is between 0.3 to 1.2 is varied in ascending order. It is found that as the inertia weight (W) is increased from 0.3, overall performance on average increases, the optimum range values for w was found to be 0.7, 0.8, 0.9. This is so because a large inertial weight makes the global search easier whereas, a small inertial weight does not improve the local search. The experiment is then carried out to determine acceleration coefficients c1, c2 using the optimum value of w. c1 = 1.5 and c2 = 1.2 gave good results. Incorporating the above-obtained values for w, c1, c2, the experiment is then repeated for varying size of the swarm (varying the number of particles in the swarm) from 1 to 1000. As the size is increased from 10 to 25 the accuracy is improved. When the size is increased from 25 to 50 and 50 to 100 particles, it led to high computational time with minor improvements in the accuracy. The experiment was run for an ANN with one hidden layer neural network of 5 neurons and for ANN with two hidden layers of two and three neurons in the first layer and the second layer respectively. It was found that the ANN with one hidden layer of 5 neurons performed better compared to ANN with multiple hidden layers. Experiment with multiple neurons and number of neurons greater than 10 did not improve the accuracy, rather it increased the computational time. [7]

TABLE I: Approximation functions

Function	**Definition**
Linear	y = x
Cubic	$y = x^3$
Sine	y = sin(x)
Tanh	y = tanh(x)
XOR	$y = x1 \oplus x2$
Complex	$y = 1.9\{1.35 + ex1 - x2 \sin[13(x1-0.6)^2]\sin[7x2]\}$

When hyper-parameters of PSO is set as of following:
W = 0.7
C1 = 1.5
C2 = 1.2
No of Iterations = 100
No. of neurons in hidden layer = 5
Particle size = 25

TABLE II: Linear Function

Activation Function	Avg. MSE
Null	0.186
Sigmoid	0.208
Hyperbolic Tangent	0.004
Cosine	0.0081
Gaussian	0.148

TABLE III: Sine Function

Activation Function	Avg. MSE
Null	0 .299
Sigmoid	0.290
Hyperbolic Tangent	0.066
Cosine	0.024
Gaussian	0.287

TABLE IV: Complex Function

Activation Function	Avg. MSE
Null	0.140
Sigmoid	0.123
Hyperbolic Tangent	0.085
Cosine	0.095
Gaussian	0.121

TABLE V: Tanh Function

Activation Function	Avg. MSE
Null	0.435
Sigmoid	0.448
Hyperbolic Tangent	0.0004
Cosine	0.035
Gaussian	0.464

TABLE VI: Cubic Function

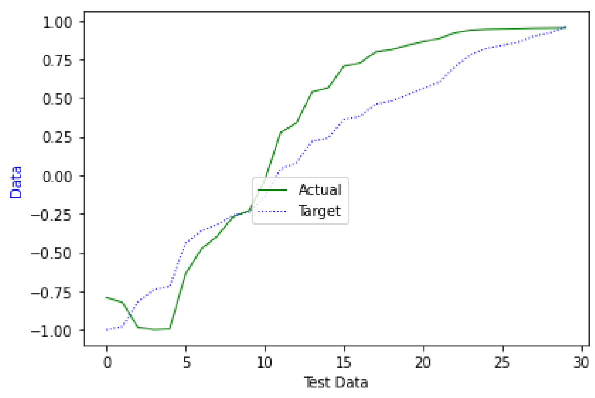
Activation Function	Avg. MSE
Null	0.073
Sigmoid	0.066
Hyperbolic Tangent	0.023
Cosine	0.026
Gaussian	0.080

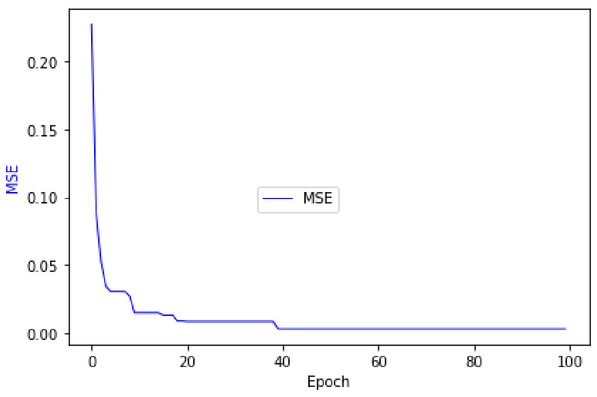
TABLE VII: XOR Function

Activation Function	Avg. MSE
Null	0.0002
Sigmoid	0.0174
Hyperbolic Tangent	0.0873
Cosine	0.0027
Gaussian	0.0001

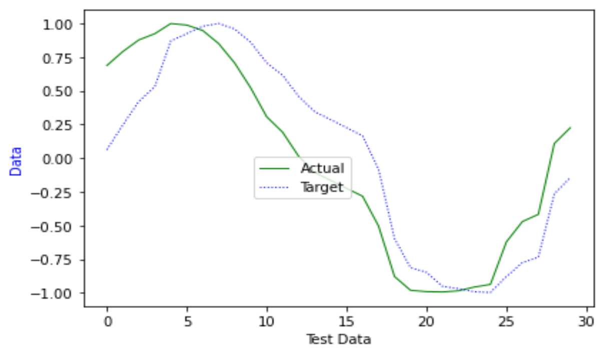
Actual- Output from MLP
Target- Targeted output from test-data
Y-axis – Output data
X-axis- Number of test data

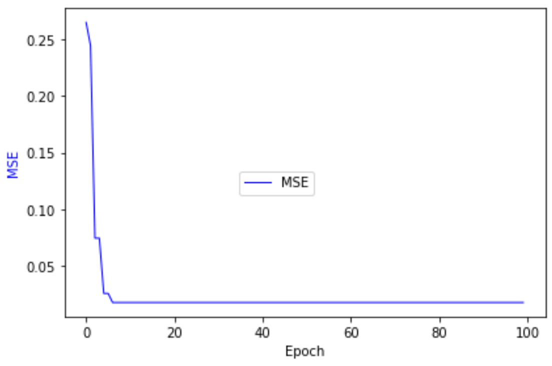
MSE of Actual vs Target plot
Y-axis – MSE
X-axis- Number of iterations



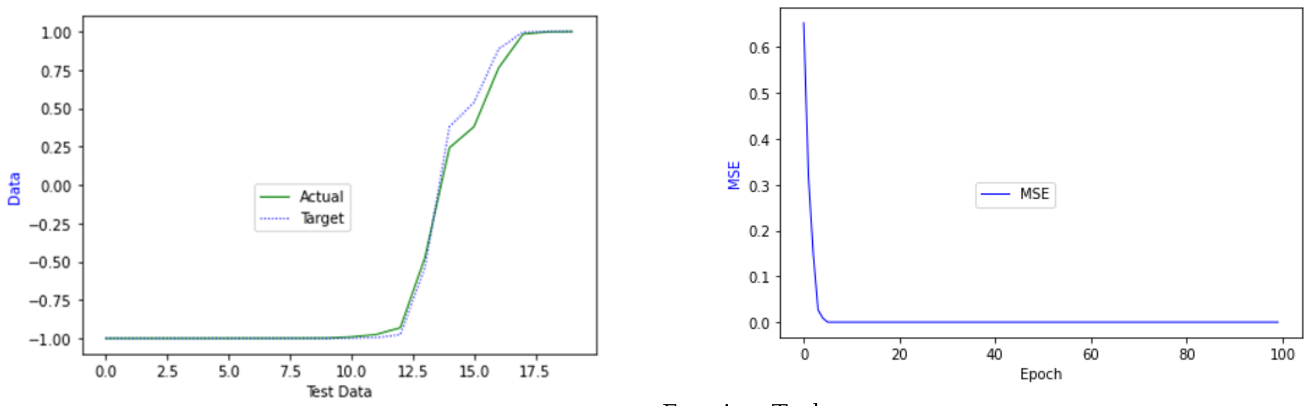


Function: Linear
Activation Function: Hyperbolic Tangent

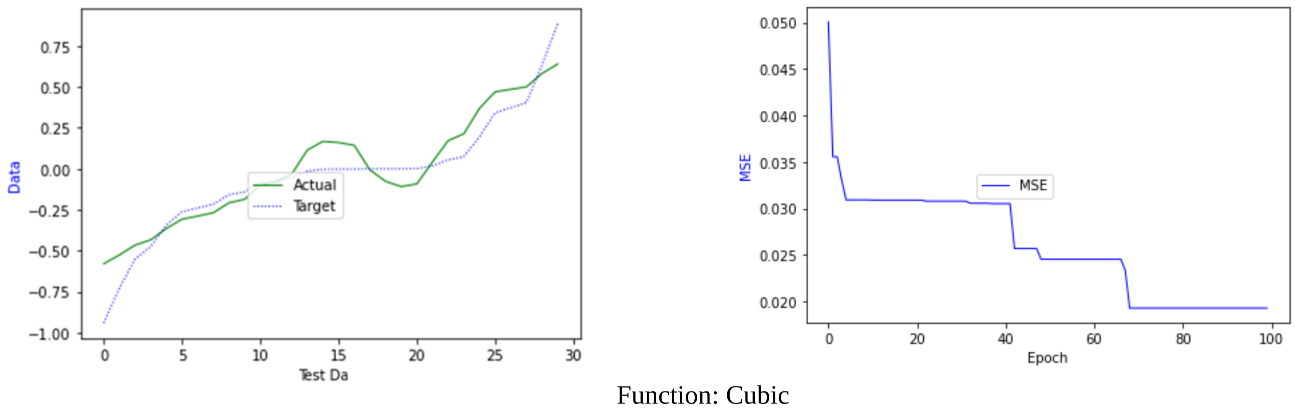




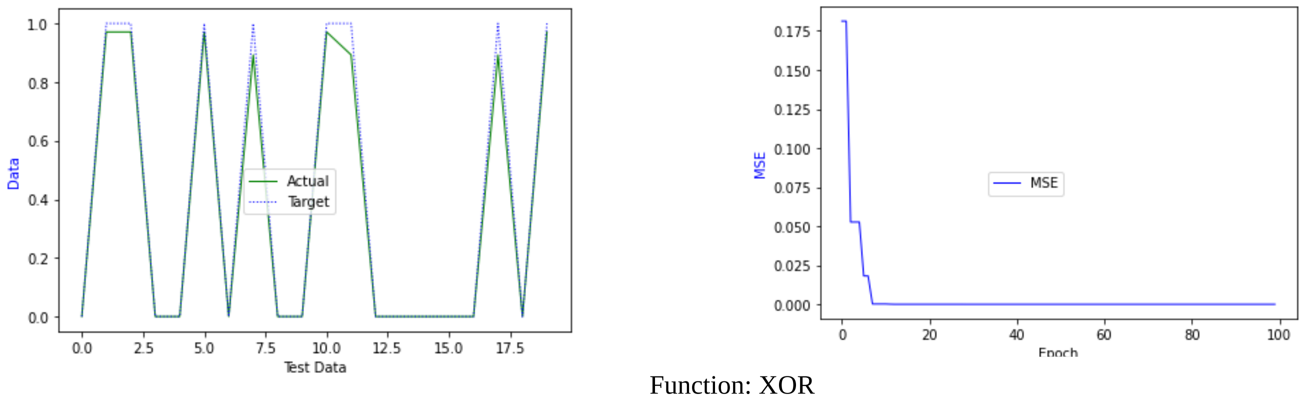
Function: Sine
Activation Function: Cosine



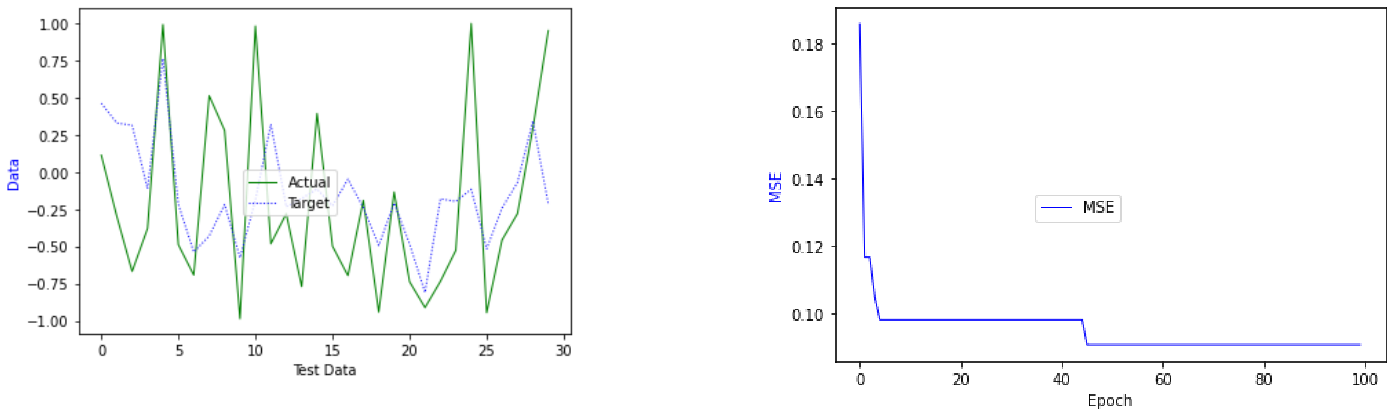
Function: Tanh
Activation Function: Hyperbolic Tangent



Function: Cubic
Activation Function: Hyperbolic Tangent



Function: XOR
Activation Function: Sigmoid



Function: Complex
Activation Function: Cosine

V. Conclusions

This paper presents the function approximation of continuous functions using a Multi-layer Feedforward Neural Network with particle swarm optimization as the training algorithm.

The experiment was carried out by varying different hyperparameters of PSO to train the neural network.

It was observed a swarm size of 25 particles, inertial weight of 0.7 and acceleration coefficients of 1.5 and 1.2 were the optimal parameters of PSO. A neural network of one hidden layer seemed to perform better when compared to a multiple hidden layered neural network.

It has been observed that activation functions play a crucial role in predicting the output.

In this study, implementation of the basic PSO is done. There are newer and improved algorithms such as the Second Generation PSO (SGPSO) that are more capable and efficient in optimizing functions.

References

[1] Cybenko, G., Approximation by Superposition of a Sigmoidal Function, Mathematics Control Signals Systems, Vol. 2, 1989, pg 303-314.

[2] Hornik, K., M. Stinchcombe & H. White, Multilayer Feedforward Networks are Universal Approximator, Neural Networks, Vol. 2, 1989, pg 359-366

[3] James Kennedy, Russell C. Eberhan, Yuhui Shi, Swarm intelligence, Morgan Kaufmann Publishers, 2001

[4] Dehuri, Ghosh, Cho, Dehuri, Satchidananda, Ghosh, Susmita and Cho, Sung-Bae (2011) Integration of swarm intelligence and artificial neural network /. Singapore: World Scientific.

[5] Beatriz A. Garro and Roberto A. Vázquez, Designing Artificial Neural Networks Using Particle Swarm Optimization Algorithms

[6] H. Khalaf Jabbar and R.Zaman Khan, Methods to avoid over-fitting and under-fitting in supervised machine learning (Comparative study). Aligarh, 2015, p.165

[7] G. Montavon, G. Orr and K. Müller, Neural networks: tricks of the trade, 2nd ed. 2012.

[8] G. K. Venayagamoorthy and V. G. Gudise, "Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks," Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003. SIS '03, Institute of Electrical and Electronics Engineers (IEEE), Jan 2003. The definitive version is available at https://doi.org/10.1109/SIS.2003.1202255

[9] H. Yoshida, Y. Fuknyama, S. Takayama. and Y. Nakanishi., "A particle swarm optimization far rcactivc power and voltagc control in ~lcclrie powcr syrlcms considcring vollagc sccurity assessment." IEEE SMC '99 Conf. Prcxcedings. Vol: 6, pp. 497 -502, I999

[10] Examining Metaheuristic Performance for Network Training. .

[11] [3:12 PM] Maheswaran, Dilani

[12] [4]Z. ZAINUDDIN and O. PAULINE, "Function Approximation Using Artificial Neural Networks", INTERNATIONAL JOURNAL OF SYSTEMS APPLICATIONS, ENGINEERING & DEVELOPMENT, vol. 1, no. 4, 2007. [Accessed 21 November 2020].

APPENDIX

Console based application written on python 3.8
IDE used Spyder

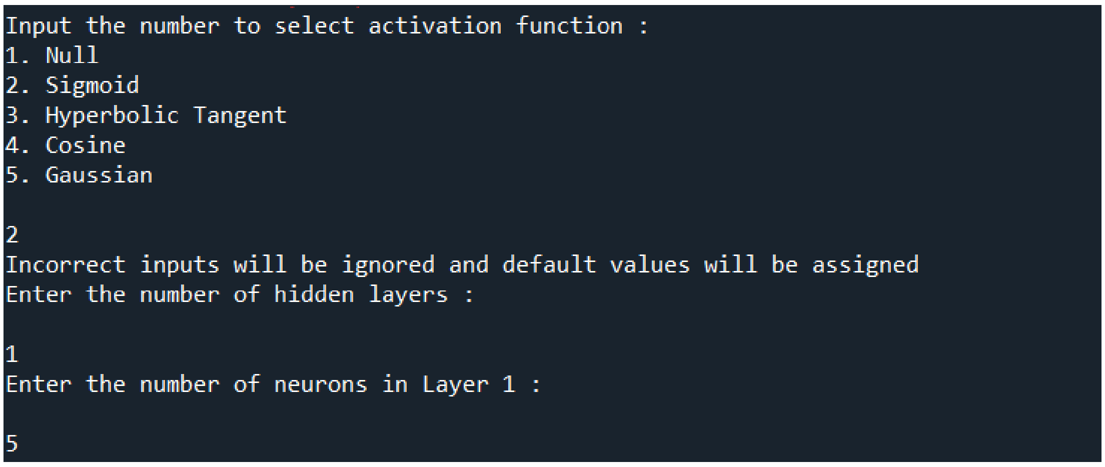
<u>Steps to run the program</u>

Program consists of three files:

- Helper.py – class for fetching user inputs and reading data sets
- ANN.py – implements feed-forward of the neural network
- PSO.py – class implements particle swarm optimization algorithm

Step-1: Run PSO.py

- Enter Activation Function
- Enter the number of Hidden layers and specify the number of neurons in the hidden layers.

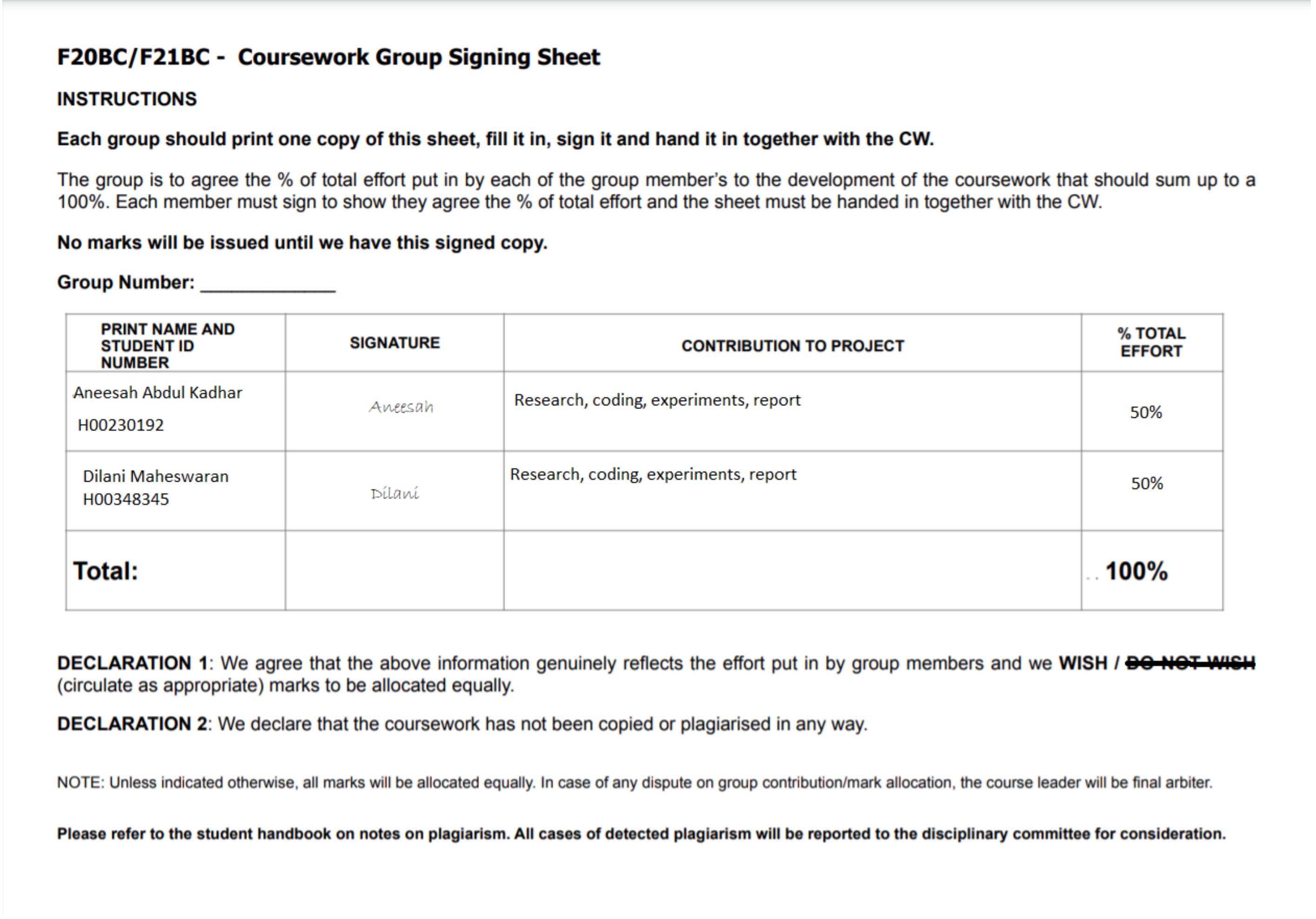


```
Input the number to select activation function :
1. Null
2. Sigmoid
3. Hyperbolic Tangent
4. Cosine
5. Gaussian

2
Incorrect inputs will be ignored and default values will be assigned
Enter the number of hidden layers :

1
Enter the number of neurons in Layer 1 :

5
```



F20BC/F21BC - Coursework Group Signing Sheet

INSTRUCTIONS

Each group should print one copy of this sheet, fill it in, sign it and hand it in together with the CW.

The group is to agree the % of total effort put in by each of the group member's to the development of the coursework that should sum up to a 100%. Each member must sign to show they agree the % of total effort and the sheet must be handed in together with the CW.

No marks will be issued until we have this signed copy.

Group Number: _____________

PRINT NAME AND STUDENT ID NUMBER	SIGNATURE	CONTRIBUTION TO PROJECT	% TOTAL EFFORT
Aneesah Abdul Kadhar H00230192	Aneesah	Research, coding, experiments, report	50%
Dilani Maheswaran H00348345	Dilani	Research, coding, experiments, report	50%
Total:			**100%**

DECLARATION 1: We agree that the above information genuinely reflects the effort put in by group members and we **WISH** / ~~**DO NOT WISH**~~ (circulate as appropriate) marks to be allocated equally.

DECLARATION 2: We declare that the coursework has not been copied or plagiarised in any way.

NOTE: Unless indicated otherwise, all marks will be allocated equally. In case of any dispute on group contribution/mark allocation, the course leader will be final arbiter.

Please refer to the student handbook on notes on plagiarism. All cases of detected plagiarism will be reported to the disciplinary committee for consideration.