HOMEWORK #4 + PROGRAMMING ASSIGNMENT #3

CSE 625

SUBMITTED BY:

UNNATI ERAMANGALATH

930001393

(QUESTION NUMBER 6,8,11,12 ATTACHED AT THE END)

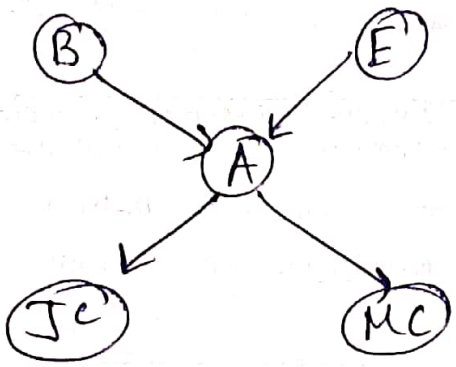
#HomeWork 4 + Program 3

UNNATI ERAMANGALATH
UIN: 930001393

1. Uncertainty and Probabilistic Reasoning

Q1. P(JohnCalls, ⌐MaryCalls, Alarm, Earthquake, ⌐Burglary)

Sol.



B = Burglary
A = Alarm
E = Earthquake
JC = John calls
MC = Mary calls.

P(B) = 0.001 P(E) = 0.002
P(-B) = 0.999 P(-E) = 0.998

B	E	P(A)	P(-A)
T	T	0.95	0.05
T	F	0.94	0.06
F	T	0.29	0.71
F	F	0.001	0.99

A	P(J)	P(-J)
T	0.90	0.10
F	0.05	0.95

A	P(M)	P(-M)
T	0.70	0.30
F	0.01	0.99

$\therefore P(J, -M, A, E, \neg B)$

$\Rightarrow P\left(\frac{J}{A}\right) \times P\left(\frac{-M}{A}\right) \times P\left(\frac{A}{B,E}\right) \times P(E) \times P(-B)$

$= 0.90 \times 0.30 \times 0.29 \times 0.002 \times 0.999$

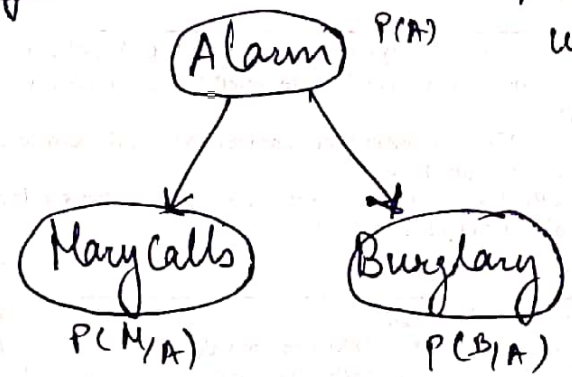
$= 1.564 \times 10^{-4}$

$= \boxed{0.0001564}$

Q2 Node Ordering – Alarm, Mary calls, Burglary.

(1) Let Alarm → A
Mary calls → M
Burglary → B

Dependency:



Alarm has a direct dependency with Mary calls and also has a direct dependency with Burglary.

(2) Node order → (A, M, B)

$\therefore \Rightarrow P(A) * P(\frac{M}{A}) * P(\frac{B}{A,M})$

Since $P(\frac{M}{A}) \neq P(M)$ and $P(\frac{B}{A}) \neq P(B)$

they both have dependencies and hence the arrow.

To check dependency between M and B, :-

$$P(\frac{B}{A,M}) = \frac{P(B, \frac{M}{A})}{P(\frac{M}{A})} = \frac{P(B, M, A)}{P(A) \cdot P(\frac{M}{A})}$$

From the dependency given in question:

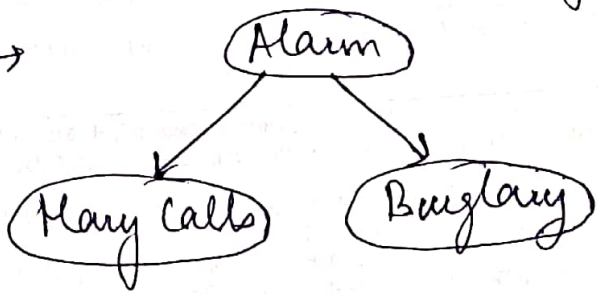
$$P(B, M, A) = P(M, B, A)$$
$$= P(\frac{M}{B,A}) * P(B, A)$$
$$= P(\frac{M}{A}) * P(B, A)$$

$$\Rightarrow P(\frac{B}{M,A}) = \frac{P(M/A) * P(B, A)}{P(A) \cdot P(M/A)} = P(B/A)$$

$\therefore P\left(\frac{B}{M,A}\right) = P\left(\frac{B}{A}\right)$

Hence B and M donot have any dependency and thus no connection in the resulting network.

$\therefore$ Network →



Q3. Explain:

P(Earthquake | Mary calls) > P(Earthquake | Mary calls, Burg)

$\Rightarrow P(E|M) > P(E|M,B)$

According to probabilistic Intercausal Inference: Causes of a common effect (explaining away: cause has already been found).

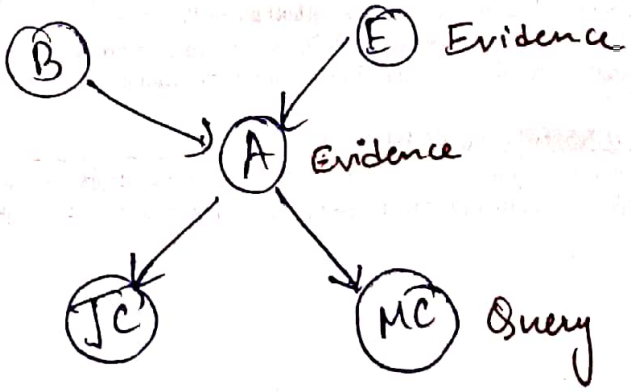
P(cause | Effect) >> P(cause | Effect $\wedge$ otherCause)

Here, ~~Eor~~ Earthquake (cause) causes an alarm, which in turn causes Mary to Call (Effect). Following the conclusion of intercausal inference, we see Burglary is an other cause. Thus:

$P(E|M) > P(E|M,B)$

Explanation:

Intercausal reasoning involves reasoning about the mutual causes of a common effect.



For our case,

B = Burglary
A = Alarm
E = Earthquake

→ Alarm can be caused either by Burglary or by an Earthquake.

→ Initially these two causes are independent.

→ Now we found evidence of "Mary calls". This new information explains the Alarm, which in turn lowers the probability that the alarm was caused by an Earthquake.

→ Even though the two causes are initially independent, with knowledge of one cause, the alternative cause is "explained away".

→ The parent node become dependent given information about the common effect. They are said to be conditionally dependent.

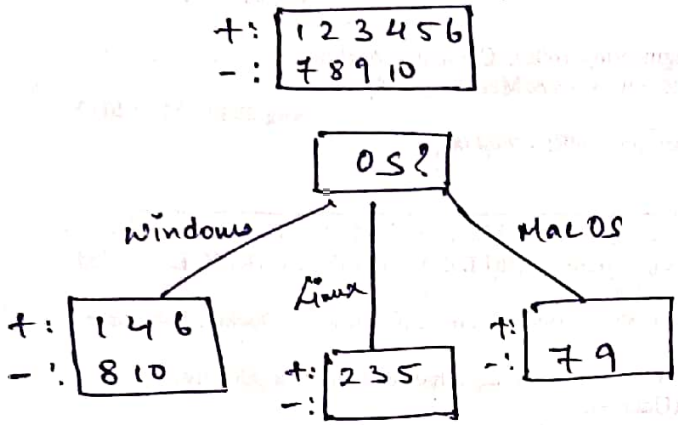
$\therefore P(E|M,B) \neq P(E|M)$

$\therefore P(E|M) > P(E|M,B)$

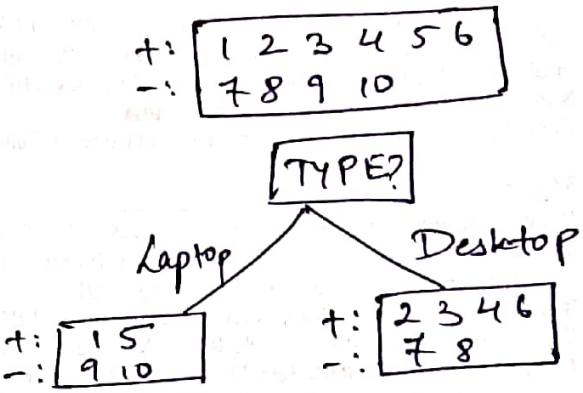
Thus explained.

Q4 (1) Depth - One Decision Tree.

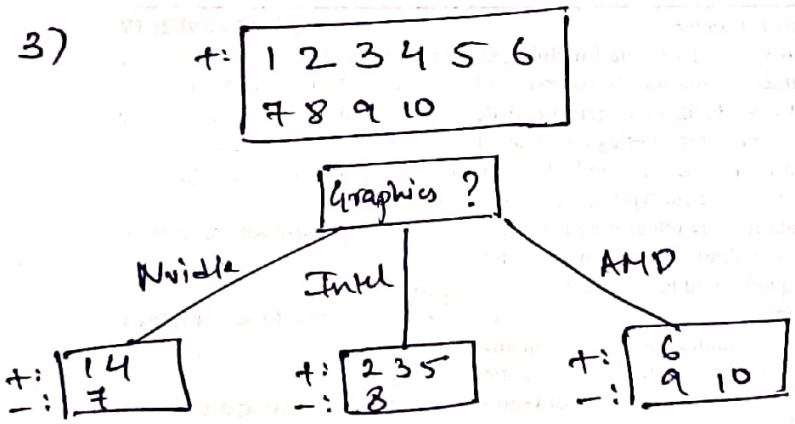
1) Attribute OS :



2) Attribute Type



3)



(2) Information Gain

(1) OS: $\text{Entropy}(E) = \sum_{i \in C} - P_i \log_2 (P_i)$

Class : $P = 6, \; N = 4$

$$\text{Entropy(class)} = \frac{-P}{P+N} \log_2 \left(\frac{P}{P+N}\right) - \frac{N}{P+N} \log_2 \left(\frac{N}{P+N}\right)$$

$$= \frac{-6}{10} \log_2 \left(\frac{6}{10}\right) - \frac{4}{10} \log_2 \left(\frac{4}{10}\right)$$

$$= 0.4416 + 0.5284$$

$$= 0.9709505$$

① OS :

	Pi	Ni	Entropy (attribute) (I)
Windows	3	2	0.97095
Linux	3	0	0
Mac	0	2	0

$$E(\text{Windows}) = -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right)$$

$$= 0.97095$$

$$\text{Entropy of OS} : \sum \frac{P_i + N_i}{P+N} (I(P_i, N_i))$$

$$= \frac{3+2}{6+4} \times 0.97 + \frac{3+0}{6+4} \times 0 + \frac{0+2}{6+4} \times 0$$

$$= \underline{0.4854}$$

$$\text{Gain}(E, A) = \text{Entropy}(E) - \sum_{v \in \text{values}(A)} \frac{|E_v|}{|E|} \text{Entropy}(E_v)$$

$$= 0.97095 - 0.4854$$

$$\boxed{IG = 0.485475} \longrightarrow \text{OS}$$

② Type is

	Pi	Ni	IGi
Laptop	2	2	1
Desktop	4	2	0.918

$$E(\text{Desktop}) = -\frac{4}{6}\log_2\left(\frac{4}{6}\right) - \frac{2}{6}\log_2\left(\frac{2}{6}\right) = \boxed{0.918295}$$

$$\text{Entropy of type} : \frac{4+2}{6+4} \times 1 + \frac{4+2}{6+4} \times 0.91825$$

$$= 0.95095$$

$$\therefore \text{Gain} = 0.97095 - 0.95095 = \boxed{0.019973}$$ Type.

③ Graphics

	Pi	Ni	Ihi
Nvidia	2	1	0.9182
Intel	3	1	0.81127
AMD	1	2	0.9182

$E(Nvidia) = -\frac{2}{3}\log_2\left(\frac{2}{3}\right) - \frac{1}{3}\log\left(\frac{1}{3}\right) = 0.9182$

$E(Intel) = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right) = 0.81127.$

$E(AMD) = -\frac{1}{3}\log_2\left(\frac{1}{3}\right) - \frac{2}{3}\log_2\left(\frac{2}{3}\right) = 0.9182$

Entropy of Graphics $= \frac{2+1}{10} \times 0.9182 + \frac{3+1}{10} \times 0.81127 + \frac{1+2}{10} \times 0.9182 = 0.8754$

$IG = 0.97095 - 0.8754$

$= \boxed{0.095461} \leftarrow$ Graphics.

③ Attribute ~~Entropy~~ Info Gain is in Descending Order:

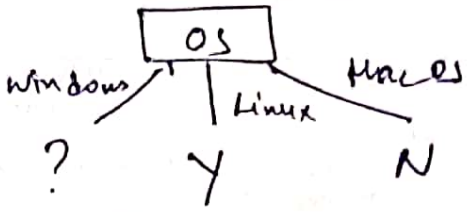
0.4854 (OS) > 0.0954 (Graphics) > 0.0199 (Type)

We will choose OS as the attribute as it has the highest value of information gain. The information gain is based on the decrease in entropy after a dataset is split on an attribute. Information gain tells which attribute is ~~more~~ most relevant, so they can be tested near the root of the tree. Information gain is the main key that is used by Decision tree algorithms to construct a decision tree.

Since IG measures how much "information" a feature gives us about the class, an attribute with highest information gain will be tested/split first. Here we choose **OS** first.

OS Decision Tree Cont...

TILL NOW:



New Table

	Type	Graphics	Decision
windows	Laptop	Nvidia	Y
windows	Desktop	Nvidia	Y
windows	Desktop	AMD	Y
windows	Desktop	Intel	N
windows	Laptop	AMD	N

Class: $P = 3, N = 2$

$$E(C) = \frac{-3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right) = \boxed{0.97095}$$

Attribute)

→ Type

	P_i	N_i	IG_i
Laptop	1	1	1
Desktop	2	1	0.91829

$$\text{Entropy of type} = \frac{2}{5} \times 1 + \frac{3}{5} \times 0.91829 = 0.9509$$

$$IG = E_{class} - E_{type} = \boxed{0.0199730}$$

↑ type.

Graphics

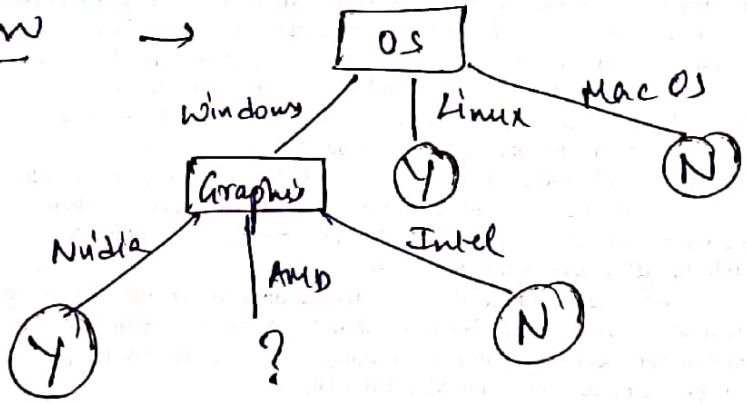
	P_i	N_i	I_{Gi}
Nvidia	2	0	0
AMD	1	1	1
Intel	0	1	0

Entropy of Graphics = $\frac{2}{5}x0 + \frac{2}{5}x1 + \frac{1}{5}x0 = 0.57$

Info. Gain of Graphics > Info gain of Type.
(0.57 > 0.01997)

Next Attribute chosen → Graphics.

Tree till Now →



New Table

	Type	Graphics	Decisions.
Windows	Desktop	AMD	Y
Windows.	Laptop	AMD	N

Class: P = 1, N = 1

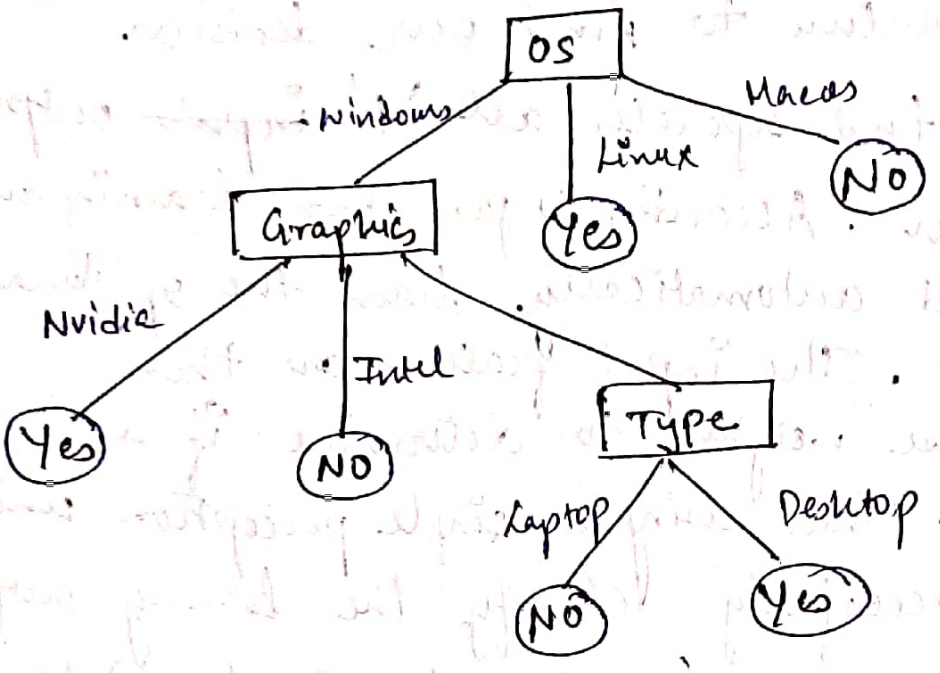
∴ E(class) = 1

	P_i	N_i	I_{Gi}
Laptop	0	1	0
Desktop	1	0	0

E(type) = 0

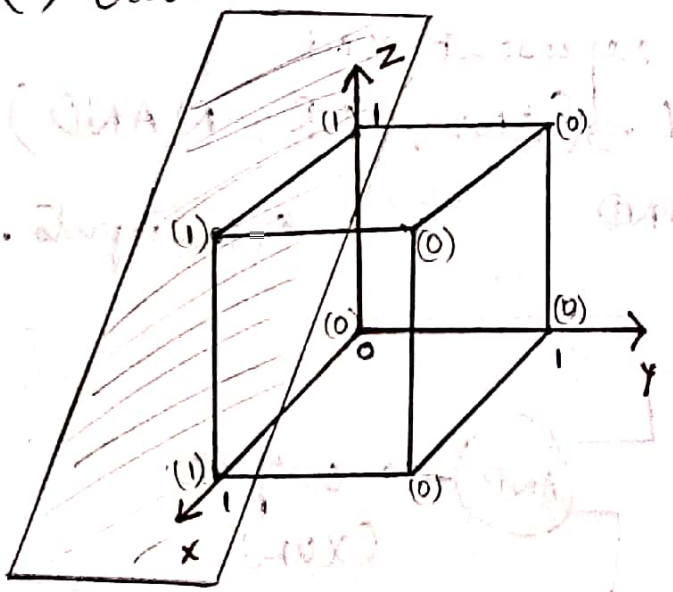
∴ Information Gain = E(class) − E(type) = 1 − 0 = 1

Final Decision Tree.



Q7 (1) YES → Single perceptron unit → Single decision Boundary (2-D plane here)

(2) Geometric Illustration:



(3) Reasoning:

→ A perceptron is a classifier. With the given set of inputs in the table it produces a class having either 0 or 1 value (binarily classified data). Plotting the values of class for the given inputs x, y and z, on a 3-D structure, we observe that all values having '1' are accumulated to the left half of the structure (see Fig).

P.T.O

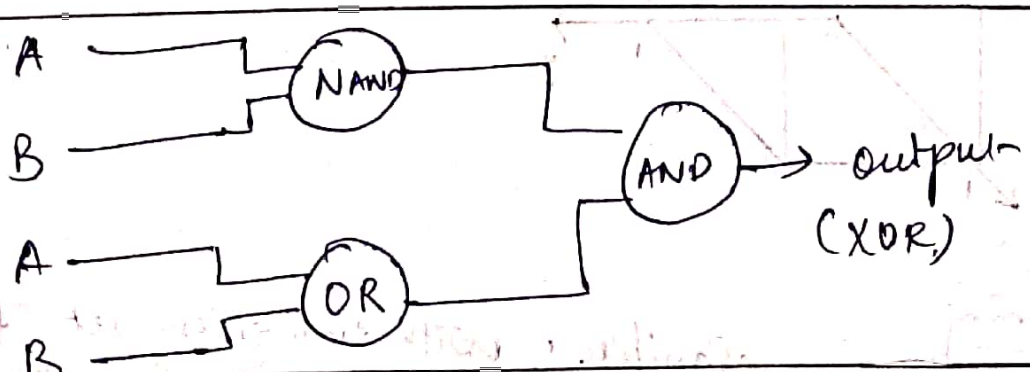
Thus we can use a 2-D plane on the left-half of 3-D structure to find our decision boundary / plane that separates all '1' ~~inputs~~ outputs from '0' outputs. According to perceptron learning rule, the algorithm would automatically learn the optimal weight co-efficients. The input features are then multiplied with these weights to determine if a neuron fires or not. Thus using a single perceptron unit, we are able to successfully classify the binary output by drawing a decision boundary (2-D plane) to separate and classify outputs.

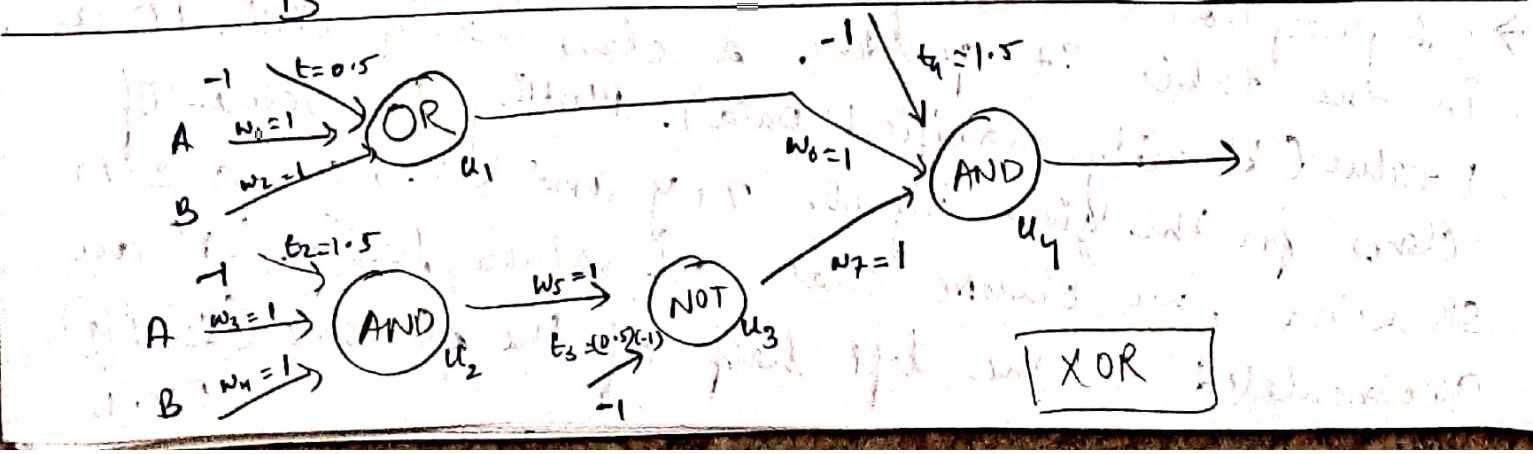
Q9 3 Perceptron ~~Using~~ & Used to Represent XOR

(Perceptron units used → (AND, OR, NAND)

AND + NOT = NAND

A, B = inputs.





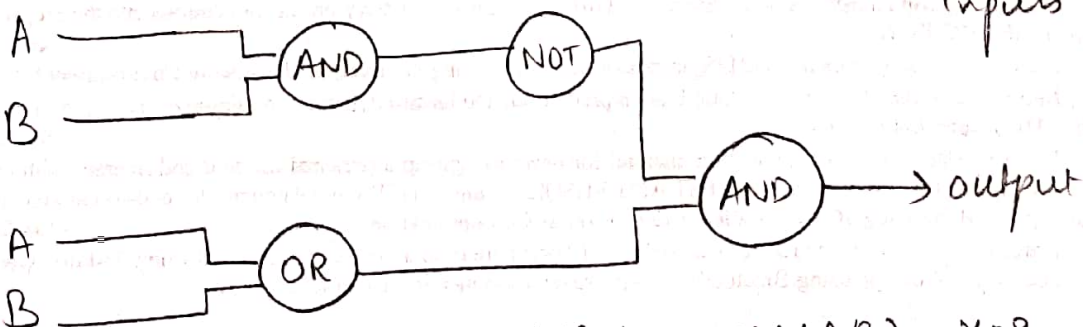
Q9 We can implement XOR function either with a combination of AND, OR and NOT units & also with 3 perceptron units.

(i) Using AND, OR, NOT :

Network Topology :

Since XOR function is not linearly separable, it can be designed in neural networks with a combination of AND, OR and NOT functions.

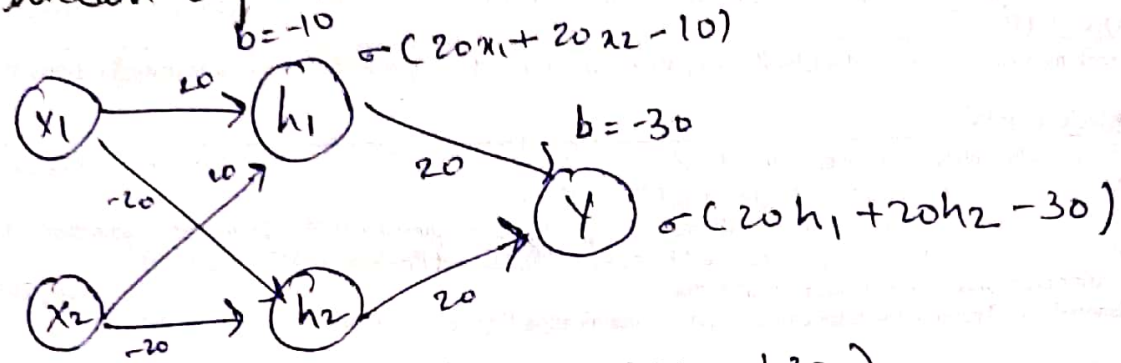
Considering these threshold values and input weight values for the given units respectively, XOR can be represented as:



where A, B are inputs.

A	B	$A \wedge B$	$\sim(A \wedge B)$	$A \vee B$	$\sim(A \wedge B) \wedge (A \vee B) = XOR$
0	0	0	1	0	0
0	1	0	1	1	1
1	0	0	1	1	1
1	1	1	0	1	0

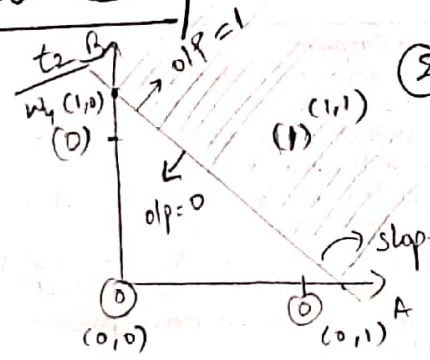
(ii) Using hidden layer.



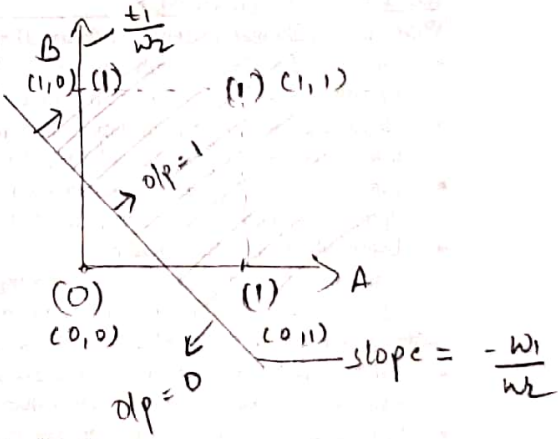
where x_1 x_2 = inputs, y = output.
h_1 h_2 = hidden layer

R) Decision Boundary

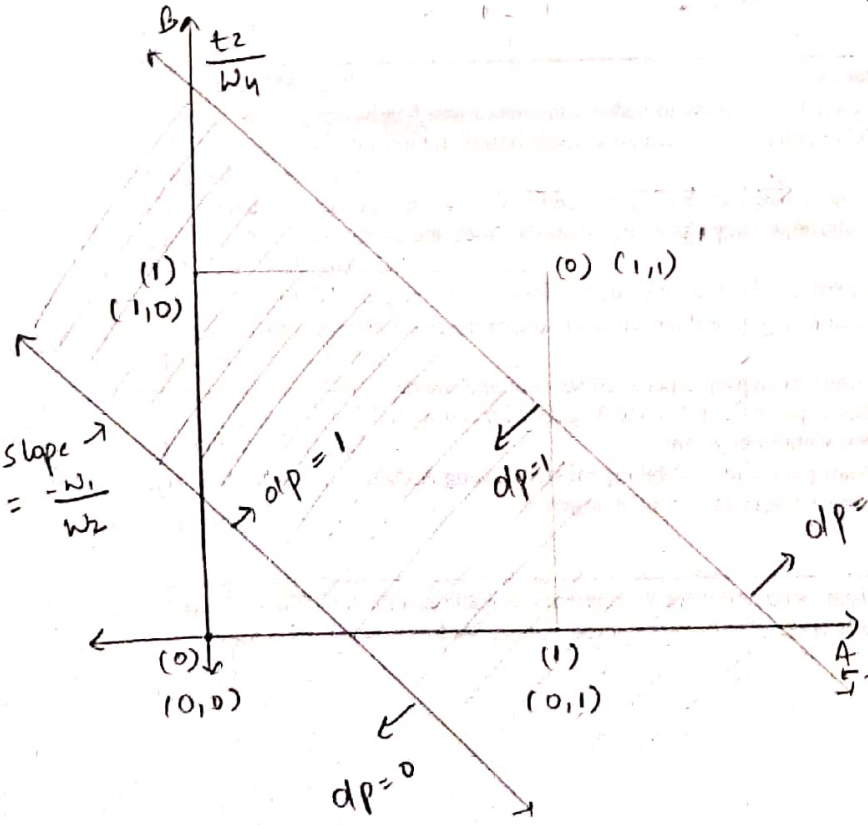
① AND :



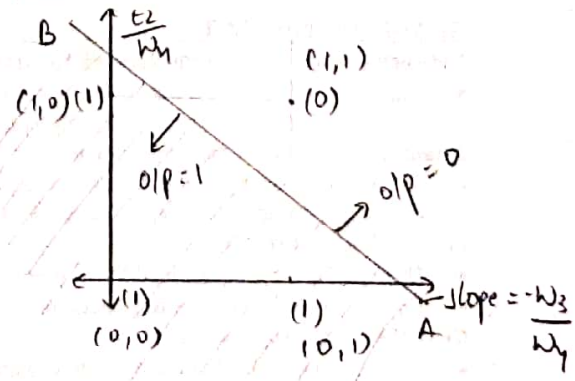
② OR:



③ XOR :



④ NAND



o/p = 0 ← (NAND + AND + OR)

(3 perceptron units)

Q10 $E(w) = (w+1)(w-1)(w-3)(w-4)$

$= (w^2-1)(w(w-4) - 3(w-4))$

$= (w^2-1)(w^2 - 7w + 12)$

$= w^2(w^2 - 7w + 12) - 1(w^2 - 7w + 12)$

$= w^4 - 7w^3 + 12w^2 - w^2 + 7w - 12$

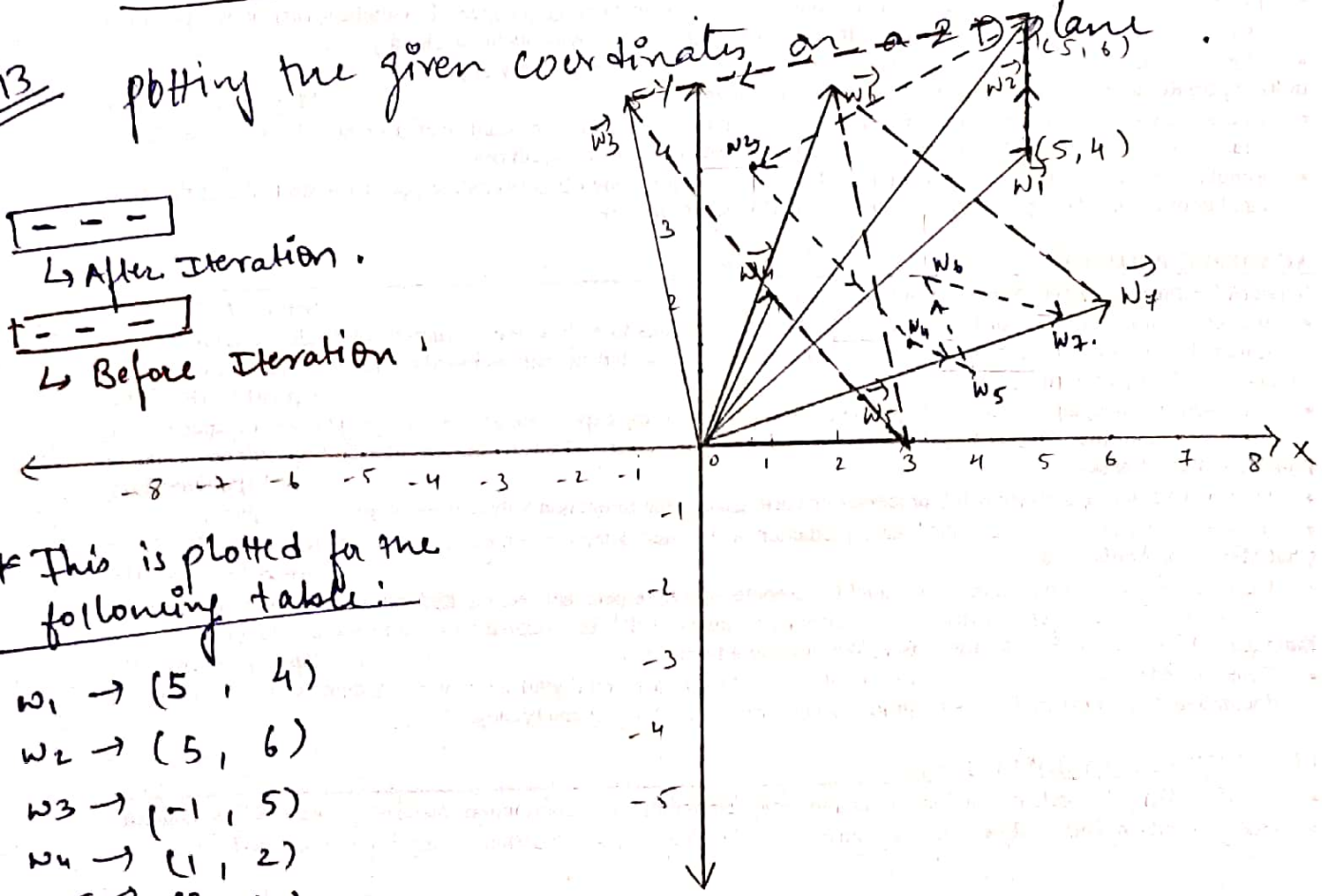
$$\Delta w = -\alpha \frac{dE}{dw}$$

$$\therefore \frac{dE}{dw} = \frac{d}{dw}(w^4 - 7w^3 + 12w^2 + 7w - 12)$$

$$= 4w^3 - 21w^2 + 22w + 7$$

$$\therefore \boxed{\Delta w = -\alpha(4w^3 + -21w^2 + 22w + 7)}$$

Q13 plotting the given coordinates on a 2D plane.



- - - → After Iteration.

- - - → Before Iteration.

* This is plotted for the following table:

$w_1 \rightarrow (5, 4)$

$w_2 \rightarrow (5, 6)$

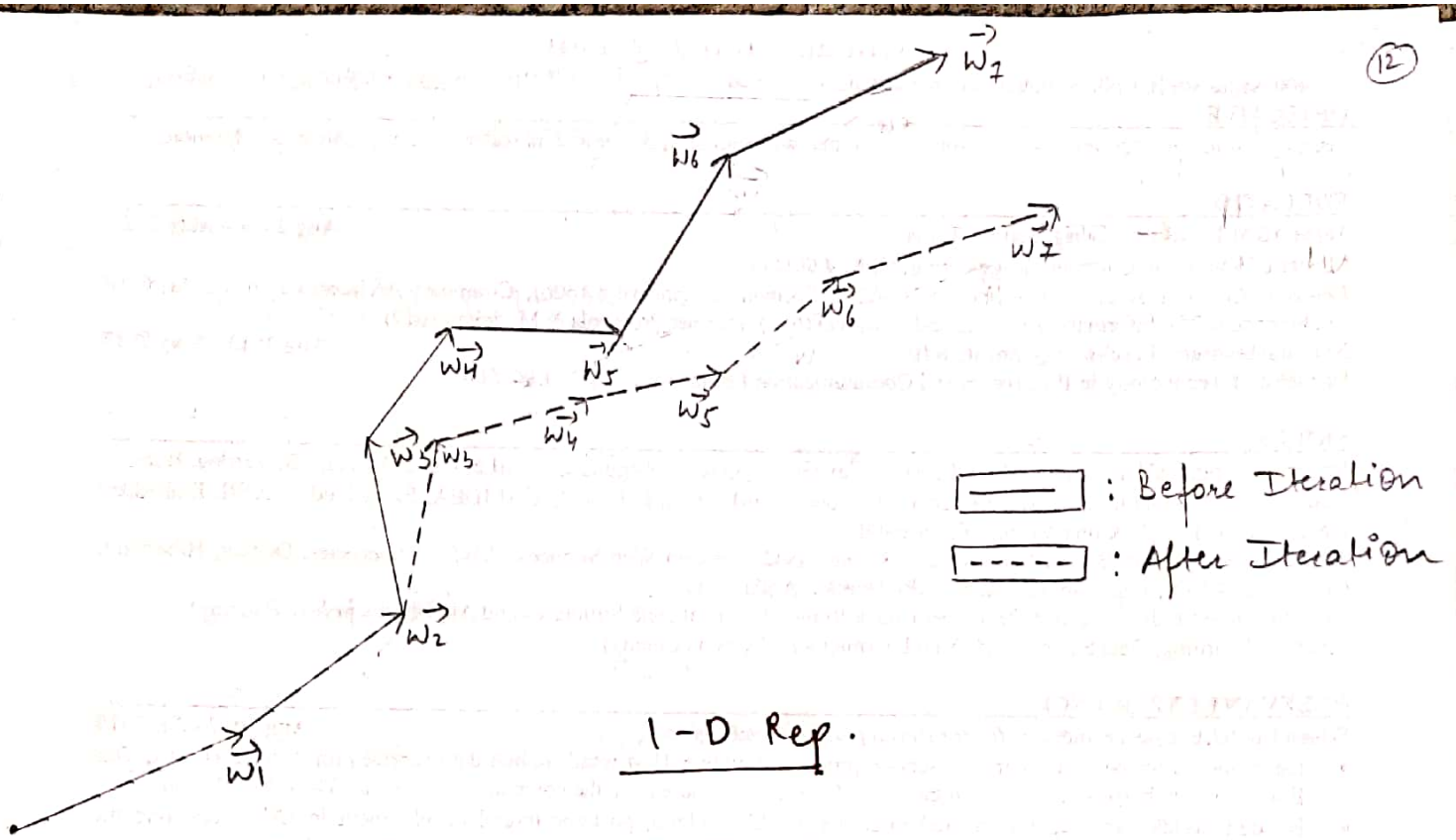
$w_3 \rightarrow (-1, 5)$

$w_4 \rightarrow (1, 2)$

$w_5 \rightarrow (3, 0)$

$w_6 \rightarrow (2, 5)$

$w_7 \rightarrow (6, 2)$



1-D Rep.

(2) $\vec{x} = (4,1)$

$$\vec{w}_j \leftarrow \vec{w}_j + \eta\, h(j, i(\vec{x}))\,(\vec{x} - \vec{w}_j)$$

where:

$h(j, i(\vec{x})) = 1 \quad \forall \; j = i(\vec{x})$

$h(j, i(\vec{x})) = \frac{2}{3} \quad \forall \; j = i(\vec{x}) \pm 1$

$h(j, i(\vec{x})) = \frac{1}{3} \quad \forall \; j = i(\vec{x}) \pm 2$

$h(j, i(\vec{x})) = 0 \quad \forall$ rest.

Shortest euclidean distance: ($\vec{x} = (4,1)$)

1) $w_1 \to (5,4)$

$\therefore d_1 = ((4-5)^2 + (1-4)^2) = 1 + 9 = 10$

2) $w_2 \to (5,6)$

$d_2 = ((4-5)^2 + (1-6)^2) = (1 + 25) = 26$

3) $w_3 \to (-1,5)$

$d_3 \to ((4+1)^2 + (1-5)^2) = 25 + 16 = 41$

4) $w_4 \to (1,2)$

$d_4 = (4-1)^2 + (1-2)^2 = 9 + 1 = 10$

5) $w_5 \rightarrow (3,0)$

$d_5 = (4-3)^2 + (1-0)^2 = 1+1 = 2.$

6) $w_6 \rightarrow (2,5)$

$d_6 = (4-2)^2 + (1-5)^2 = 4+16 = 20$

7) $w_7 \rightarrow (6,2)$

$d_7 = (4-6)^2 + (1-2)^2 = 4+1 = 5$

* w_5 is having the shortest distance to input vector $\vec{x}$ among all other vectors.

So for 1st iteration we choose w_5.

	w_1	w_2	w_3	w_4	w_5	w_6	w_7
$h(j, i(\vec{x}))$	0	0	$\frac{1}{3}$	$\frac{2}{3}$	1	$\frac{2}{3}$	$\frac{1}{3}$

∴ Change in vector weights: $(\alpha = \eta = 1)$

1) $w_1 \leftarrow w_1 + 1 \times 0 \times ((4,1) - (5,4))$

$\boxed{w_1 = (5,4)}$

2) $w_2 \leftarrow (5,6) + 1 \times 0 \times ((4,1) - (5,6))$

$\boxed{w_2 = (5,6)}$

3) $w_3 \leftarrow (-1,5) + 1 \times \frac{1}{3} \times (-1,-5)$

$\boxed{w_3 = \left(\frac{-4}{3}, \frac{10}{3}\right)} = \cancel{(-1.33, 3.33)} \quad \boxed{(0.66, 3.66)}$

4) $w_4 \leftarrow (1,2) + 1 \times \frac{2}{3} \times ((4,1) - (1,2))$

$= (1,2) + \frac{2}{3}(3,-1)$

$= (3, \frac{4}{3}) = (3, 1.33)$

5. $W_5 \leftarrow (3,0) + 1 \times 1 \times ((4,1) - (3,0))$

$\leftarrow (3,0) + (1,1)$

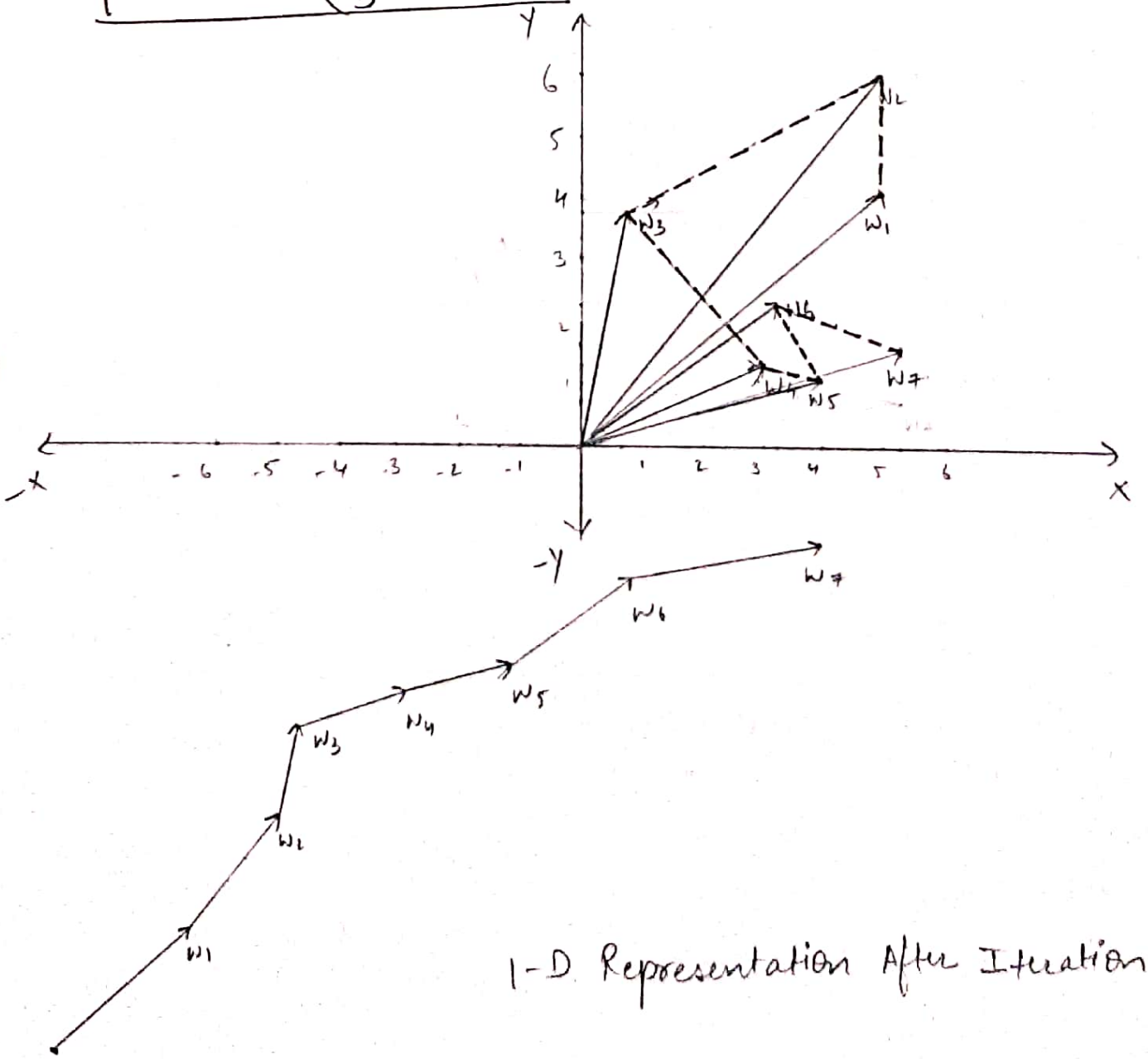
$\boxed{W_5 = (4,1)}$

6) $W_6 \leftarrow (2,5) + 1 \times \frac{2}{3} \times ((4,1) - (2,5))$

$\boxed{W_6 = \left(\frac{10}{3}, \frac{7}{3}\right)} \Rightarrow (3.33, 2.33)$

7) $W_7 \leftarrow (6,2) + 1 \times \frac{1}{3} \times ((4,1) - (6,2))$

$\boxed{W_7 = \left(\frac{16}{3}, \frac{5}{3}\right)} = (5.33, 1.66)$



1-D Representation After Iteration.

6. **INFORMATION GAIN**

CODE (PYTHON LANGUAGE)

```
from math import log2

#Calculates Entropy
def cal_entropy(positive, negative):
    if(positive == negative):
        return 1
    if(positive == 0 or negative ==0):
        return 0
    else:
        sum_pn = positive + negative
        att1 = positive/sum_pn
        att2 = negative/sum_pn
        entropy = (-1*att1*log2(att1) + -1*att2*log2(att2))
        return entropy

#Calculate Information Gain
def infogain(pnclass, pnattributes):
    class_positive = pnclass[0]  #6
    class_negative = pnclass[1]  #6
    class_entropy = cal_entropy(class_positive,class_negative) #1
    info_gain_attri = []
    numerator = []
    denominator = class_positive + class_negative
    attribute_entropy = 0

    for i in range(len(pnattributes)):
        ig = cal_entropy(pnattributes[i][0],pnattributes[i][1])
        info_gain_attri.append(ig)
        num_sum = pnattributes[i][0] + pnattributes[i][1]
        numerator.append(num_sum)
        attribute_entropy += (((numerator[i])/denominator)*(info_gain_attri[i]))

    infogain = class_entropy - attribute_entropy

    return infogain

prior_testing_samples = [6,4]
post_testing_Samples_OS = [[3,2], [3,0],[0,2]]
post_testing_Samples_type = [[2,2], [4,2]]
post_testing_Samples_graphics = [[2,1], [3,1],[1,2]]
testing_sample_2 = [3,2]
type_sample_2 = [[1,1],[2,1]]
graphics_sample_2 = [[2,0],[1,1],[0,1]]
testing_sample_3 = [1,1]
```

```
type_sample_3 = [[0,1],[1,0]]

print("FIRST ITERATION")
print("====================")
information_gain_os = infogain(prior_testing_samples,post_testing_Samples_OS)
print("Information gain of OS attribute is :" , information_gain_os)
print()
information_gain_ty = infogain(prior_testing_samples,post_testing_Samples_type)
print("Information gain of TYPE attribute is :" , information_gain_ty)
print()
information_gain_gp = infogain(prior_testing_samples,post_testing_Samples_graphics)
print("Information gain of GRAPHICS attribute is :" , information_gain_gp)
print()
print("SECOND ITERATION")
print("====================")
information_gain_TYPE = infogain(testing_sample_2,type_sample_2)
print("Information gain of TYPE attribute (2)_ is :" , information_gain_TYPE)
print()
information_gain_Graphics = infogain(testing_sample_2,graphics_sample_2)
print("Information gain of GRAPHICS attribute (2)_ is :" , information_gain_Graphics)
print()
print("THIRD ITERATION")
print("====================")
information_gain_TYPE_3 = infogain(testing_sample_3,type_sample_3)
print("Information gain of TYPE attribute (3)_ is :" , information_gain_TYPE_3)
print()

print("-----------------------------------------------------------------------------------------")
print()
# FOR ALL ATTRIBUTES OF SLIDE 6 PAGE 9

prior_testing_goal = [6,6]
post_testing_Samples_ALT = [[3,3], [3,3]]
post_testing_Samples_BAR = [[3,3], [3,3]]
post_testing_Samples_FRI = [[2,3], [4,3]]
post_testing_Samples_HUN = [[5,2], [1,4]]
post_testing_Samples_PAT = [[4,0], [2,4],[0,2]]
post_testing_Samples_PRICE = [[3,4], [2,0],[1,2]]
post_testing_Samples_RAIN = [[2,2], [4,4]]
post_testing_Samples_RES = [[3,2], [3,4]]
post_testing_Samples_TYPE = [[1,1], [2,2],[2,2],[1,1]]
post_testing_Samples_EST = [[4,2], [1,1],[1,1],[0,2]]

print("INFORMATION GAIN FOR ALL ATTRIBUTES IN TABLE OF SLIDE 6 PAGE 9")
print("============================================================")
information_gain_alt = infogain(prior_testing_goal,post_testing_Samples_ALT)
print("Information gain of ALT attribute is :" , information_gain_alt)
```

```
print()
information_gain_bar = infogain(prior_testing_goal,post_testing_Samples_BAR)
print("Information gain of BAR attribute is :" , information_gain_bar)
print()
information_gain_fri = infogain(prior_testing_goal,post_testing_Samples_FRI)
print("Information gain of FRI attribute is :" , information_gain_fri)
print()
information_gain_hun = infogain(prior_testing_goal,post_testing_Samples_HUN)
print("Information gain of HUN attribute is :" , information_gain_hun)
print()
information_gain_pat = infogain(prior_testing_goal,post_testing_Samples_PAT)
print("Information gain of PAT attribute is :" , information_gain_pat)
print()
information_gain_price = infogain(prior_testing_goal,post_testing_Samples_PRICE)
print("Information gain of PRICE attribute is :" , information_gain_price)
print()
information_gain_rain = infogain(prior_testing_goal,post_testing_Samples_RAIN)
print("Information gain of RAIN attribute is :" , information_gain_rain)
print()
information_gain_res = infogain(prior_testing_goal,post_testing_Samples_RES)
print("Information gain of RES attribute is :" , information_gain_res)
print()
information_gain_type = infogain(prior_testing_goal,post_testing_Samples_TYPE)
print("Information gain of TYPE attribute is :" , information_gain_type)
print()
information_gain_est = infogain(prior_testing_goal,post_testing_Samples_EST)
print("Information gain of EST attribute is :" , information_gain_est)
print()
```

RESULT

```
FIRST ITERATION
====================
Information gain of OS attribute is : 0.4854752972273343

Information gain of TYPE attribute is : 0.01997309402197489

Information gain of GRAPHICS attribute is : 0.09546184423832171

SECOND ITERATION
====================
Information gain of TYPE attribute (2)_ is : 0.01997309402197489

Information gain of GRAPHICS attribute (2)_ is : 0.5709505944546686

THIRD ITERATION
====================
Information gain of TYPE attribute (3)_ is : 1.0

--------------------------------------------------------------------------------

INFORMATION GAIN FOR ALL ATTRIBUTES IN TABLE OF SLIDE 6 PAGE 9
================================================================
Information gain of ALT attribute is : 0.0

Information gain of BAR attribute is : 0.0

Information gain of FRI attribute is : 0.020720839623907805

Information gain of HUN attribute is : 0.19570962879973086

Information gain of PAT attribute is : 0.5408520829727552

Information gain of PRICE attribute is : 0.19570962879973075

Information gain of RAIN attribute is : 0.0

Information gain of RES attribute is : 0.020720839623907805

Information gain of TYPE attribute is : 1.1102230246251565e-16

Information gain of EST attribute is : 0.20751874963942196
```

8. **PERCEPTRON PROGRAM**

CODE

```
from random import choice
from numpy import array, dot, random, linspace
import matplotlib.pyplot as plt

unit_step_function = lambda x: 0 if x <=0  else 1

training_data_OR = [
    (array([0,0,-1]), 0),
    (array([0,1,-1]), 1),
    (array([1,0,-1]), 1),
    (array([1,1,-1]), 1),
]

training_data_AND = [
    (array([0,0,-1]), 0),
    (array([0,1,-1]), 0),
    (array([1,0,-1]), 0),
    (array([1,1,-1]), 1),
]

training_data_XOR = [
    (array([0,0,-1]), 0),
    (array([0,1,-1]), 1),
    (array([1,0,-1]), 1),
    (array([1,1,-1]), 0),
]


eta = 0.2       #Learning Rate
n = 10000000000000
maxerr = .000001    #Minimum error allowed
error = 100000

input_list = []
output_val = []
count=1

def perceptron(training_function):
    w = random.rand(3)  #initializing random weights for w
#w = [0.289727, 0.78754878, 0.87956895] --without training data for AND
#w = [0.97452233 ,1.00040192 ,0.79389944]   --- without trainig  data for OR

    for i in range(n):
        count=0
```

```
        for j in range(4):
            x, expected = training_function[j]
            result = dot(w, x)   #w1*x1 + w2*x2 + w3*x3
            error = expected - unit_step_function(result)
            print('Before updation w value : ', w)
            print('Input value : ',x)
            print('Expected output : ', expected)
            print('Returned output : ',result)
            print('Error observed : ',error )
            w += eta * error * x
            count += abs(error)
            print('After Updation w value : ', w)
        plt.plot(linspace(-1,3,100),(-w[0]/w[1])*linspace(-1,3,100)+(w[2]/w[1]), color = 'indigo')
        plt.title("DECISION BOUNDARY FOR THE GIVEN TRAINING DATA", color = 'navy')
        plt.scatter([0,0,1,1],[0,1,0,1],s=10, color = 'maroon')
        ax = plt.gca()
        ax.set_facecolor('peachpuff')
        plt.show()
        if(count == 0):
            print('BREAKING OFF')
            break
        print('updated', error * x)

        print()
        print('sum of errors', count)

perceptron(training_data_OR)
perceptron(training_data_AND)
perceptron(training_data_XOR)
```

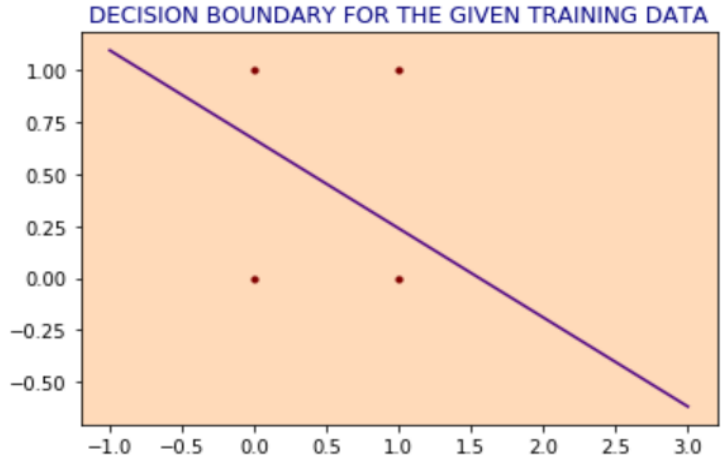
RESULTS

1. OR FUNCTION

1.RANDOM INITIAL WEIGHT - [0.20304645 , 0.9385026, 0.8268018]

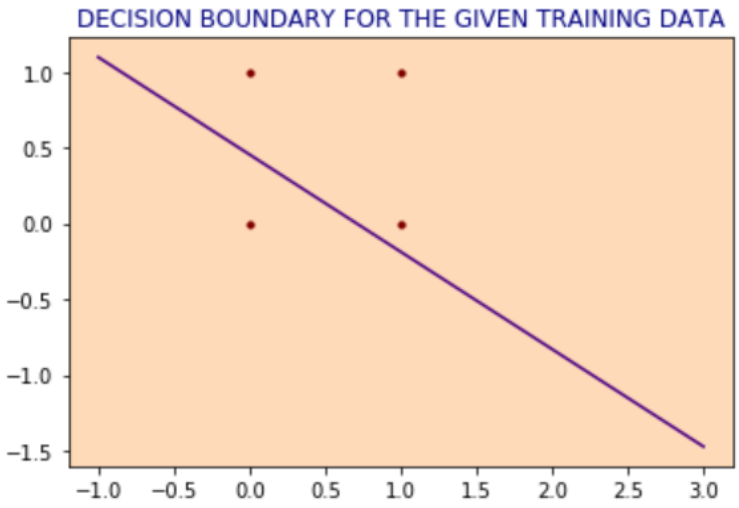
(FIRST EXAMPLE SHOWS ALL PLOTS FOR REACHING FINAL DECISION BOUNDARY)

```
Before updation w value :  [0.20304645 0.9385026  0.8268018 ]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.8268017998040822
Error observed :  0
After Updation w value :  [0.20304645 0.9385026  0.8268018 ]
Before updation w value :  [0.20304645 0.9385026  0.8268018 ]
Input value :  [ 0  1 -1]
Expected output :  1
Returned output :  0.11170080059007836
Error observed :  0
After Updation w value :  [0.20304645 0.9385026  0.8268018 ]
Before updation w value :  [0.20304645 0.9385026  0.8268018 ]
Input value :  [ 1  0 -1]
Expected output :  1
Returned output :  -0.6237553519006018
Error observed :  1
After Updation w value :  [0.40304645 0.9385026  0.6268018 ]
Before updation w value :  [0.40304645 0.9385026  0.6268018 ]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  0.7147472484935586
Error observed :  0
After Updation w value :  [0.40304645 0.9385026  0.6268018 ]
```



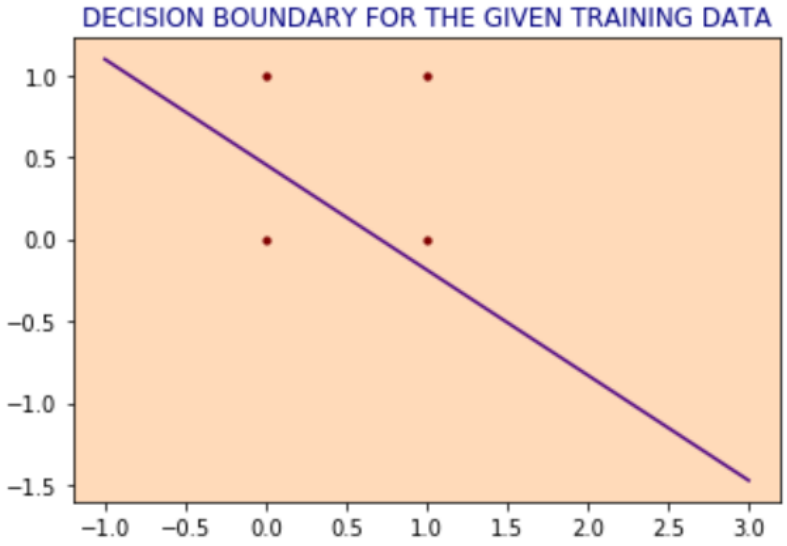

```
updated [0 0 0]

sum of errors 1
```

```
updated [0 0 0]

sum of errors 1
Before updation w value :  [0.40304645 0.9385026  0.6268018 ]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.6268017998040822
Error observed :  0
After Updation w value :  [0.40304645 0.9385026  0.6268018 ]
Before updation w value :  [0.40304645 0.9385026  0.6268018 ]
Input value :  [ 0  1 -1]
Expected output :  1
Returned output :  0.3117008005900783
Error observed :  0
After Updation w value :  [0.40304645 0.9385026  0.6268018 ]
Before updation w value :  [0.40304645 0.9385026  0.6268018 ]
Input value :  [ 1  0 -1]
Expected output :  1
Returned output :  -0.22375535190060186
Error observed :  1
After Updation w value :  [0.60304645 0.9385026  0.4268018 ]
Before updation w value :  [0.60304645 0.9385026  0.4268018 ]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  1.1147472484935586
Error observed :  0
After Updation w value :  [0.60304645 0.9385026  0.4268018 ]
```




```
updated [0 0 0]
```

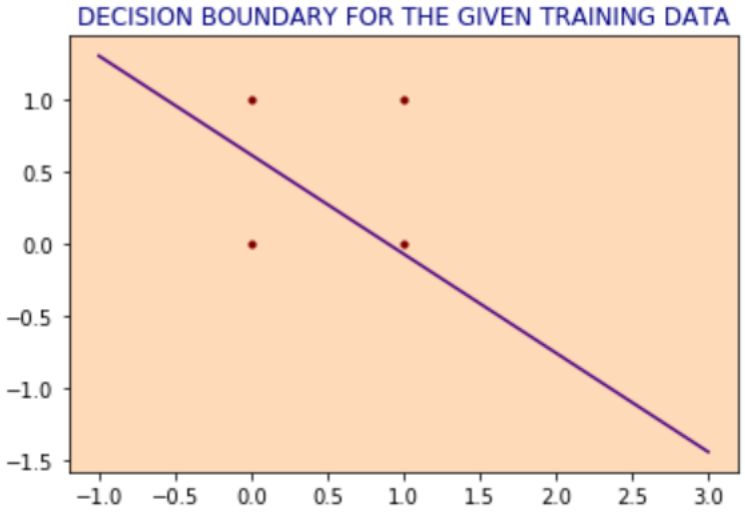
```
updated [0 0 0]

sum of errors 1
Before updation w value :  [0.60304645 0.9385026  0.4268018 ]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.4268017998040822
Error observed :  0
After Updation w value :  [0.60304645 0.9385026  0.4268018 ]
Before updation w value :  [0.60304645 0.9385026  0.4268018 ]
Input value :  [ 0  1 -1]
Expected output :  1
Returned output :  0.5117008005900783
Error observed :  0
After Updation w value :  [0.60304645 0.9385026  0.4268018 ]
Before updation w value :  [0.60304645 0.9385026  0.4268018 ]
Input value :  [ 1  0 -1]
Expected output :  1
Returned output :  0.17624464809939816
Error observed :  0
After Updation w value :  [0.60304645 0.9385026  0.4268018 ]
Before updation w value :  [0.60304645 0.9385026  0.4268018 ]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  1.1147472484935586
Error observed :  0
After Updation w value :  [0.60304645 0.9385026  0.4268018 ]
```




```
BREAKING OFF
```

*2.RANDOM INITIAL WEIGHT - [*0.25371406 0.46036573 0.81045166*]*

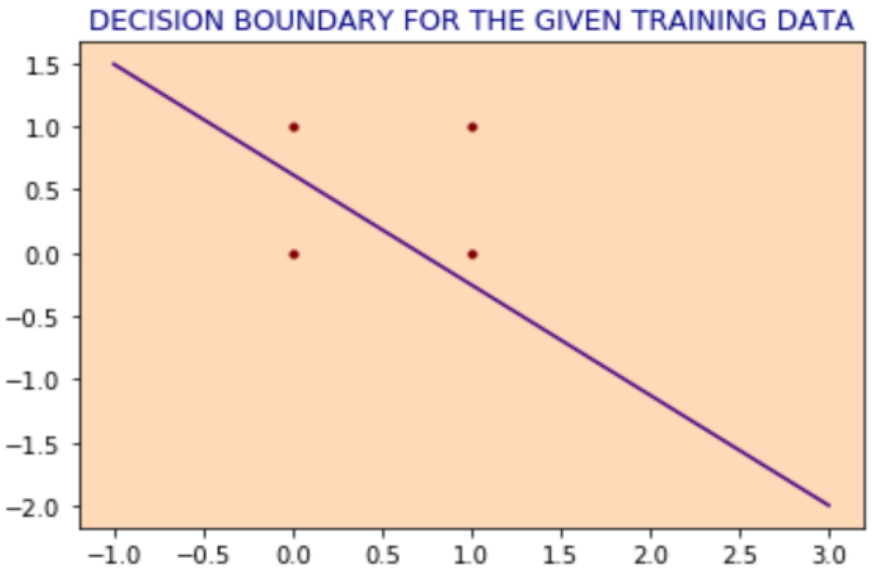
```
updated [0 0 0]

sum of errors 2
Before updation w value :  [0.45371406 0.66036573 0.41045166]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.4104516593110667
Error observed :  0
After Updation w value :  [0.45371406 0.66036573 0.41045166]
Before updation w value :  [0.45371406 0.66036573 0.41045166]
Input value :  [ 0  1 -1]
Expected output :  1
Returned output :  0.24991406577555925
Error observed :  0
After Updation w value :  [0.45371406 0.66036573 0.41045166]
Before updation w value :  [0.45371406 0.66036573 0.41045166]
Input value :  [ 1  0 -1]
Expected output :  1
Returned output :  0.04326239977260238
Error observed :  0
After Updation w value :  [0.45371406 0.66036573 0.41045166]
Before updation w value :  [0.45371406 0.66036573 0.41045166]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  0.7036281248592282
Error observed :  0
After Updation w value :  [0.45371406 0.66036573 0.41045166]
```




```
BREAKING OFF
```

*3.RANDOM INITIAL WEIGHT - [*0.66469456 0.76244415 0.47159556]

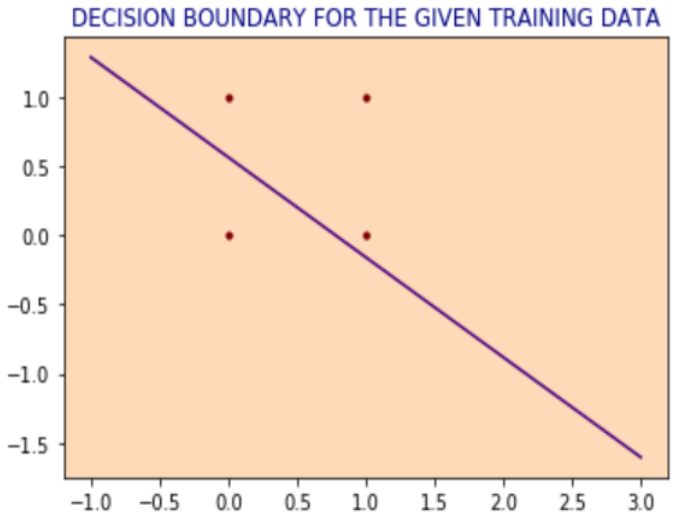
```
Before updation w value :  [0.66469456 0.76244415 0.47159556]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.47159555786943064
Error observed :  0
After Updation w value :  [0.66469456 0.76244415 0.47159556]
Before updation w value :  [0.66469456 0.76244415 0.47159556]
Input value :  [ 0  1 -1]
Expected output :  1
Returned output :  0.29084859579678923
Error observed :  0
After Updation w value :  [0.66469456 0.76244415 0.47159556]
Before updation w value :  [0.66469456 0.76244415 0.47159556]
Input value :  [ 1  0 -1]
Expected output :  1
Returned output :  0.19309899971830646
Error observed :  0
After Updation w value :  [0.66469456 0.76244415 0.47159556]
Before updation w value :  [0.66469456 0.76244415 0.47159556]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  0.9555431533845262
Error observed :  0
After Updation w value :  [0.66469456 0.76244415 0.47159556]
```




```
BREAKING OFF
```

*4.RANDOM INITIAL WEIGHT - [*0.14077693 0.47109507 0.4657787]

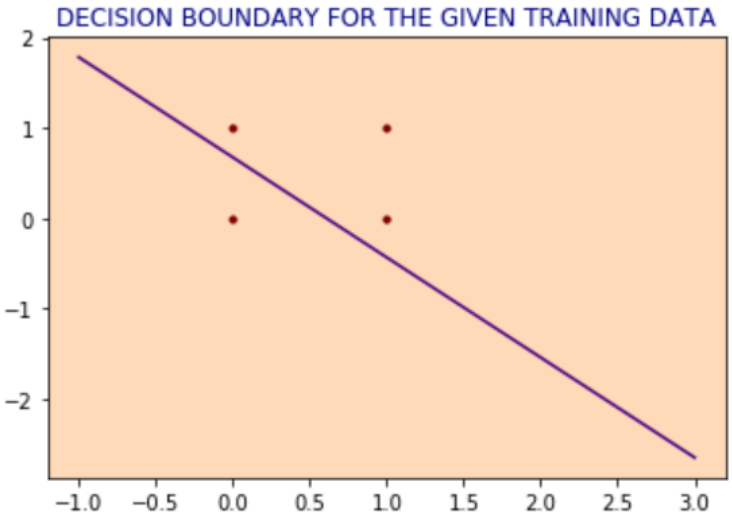
```
Before updation w value :  [0.34077693 0.47109507 0.2657787 ]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.2657787008997083
Error observed :  0
After Updation w value :  [0.34077693 0.47109507 0.2657787 ]
Before updation w value :  [0.34077693 0.47109507 0.2657787 ]
Input value :  [ 0  1 -1]
Expected output :  1
Returned output :  0.20531637015625165
Error observed :  0
After Updation w value :  [0.34077693 0.47109507 0.2657787 ]
Before updation w value :  [0.34077693 0.47109507 0.2657787 ]
Input value :  [ 1  0 -1]
Expected output :  1
Returned output :  0.07499823141378603
Error observed :  0
After Updation w value :  [0.34077693 0.47109507 0.2657787 ]
Before updation w value :  [0.34077693 0.47109507 0.2657787 ]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  0.5460933024697459
Error observed :  0
After Updation w value :  [0.34077693 0.47109507 0.2657787 ]
```




```
BREAKING OFF
```

*5.RANDOM INITIAL WEIGHT - [*0.72091807 0.65052983 0.43992476]

```
Before updation w value :  [0.72091807 0.65052983 0.43992476]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.4399247623860778
Error observed :  0
After Updation w value :  [0.72091807 0.65052983 0.43992476]
Before updation w value :  [0.72091807 0.65052983 0.43992476]
Input value :  [ 0  1 -1]
Expected output :  1
Returned output :  0.21060506818467883
Error observed :  0
After Updation w value :  [0.72091807 0.65052983 0.43992476]
Before updation w value :  [0.72091807 0.65052983 0.43992476]
Input value :  [ 1  0 -1]
Expected output :  1
Returned output :  0.28099330818474
Error observed :  0
After Updation w value :  [0.72091807 0.65052983 0.43992476]
Before updation w value :  [0.72091807 0.65052983 0.43992476]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  0.9315231387554967
Error observed :  0
After Updation w value :  [0.72091807 0.65052983 0.43992476]
```



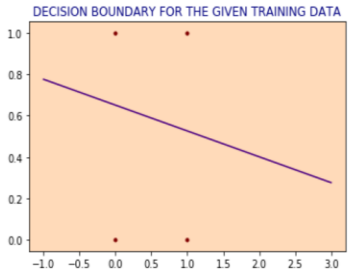

```
BREAKING OFF
```

2. AND FUNCTION

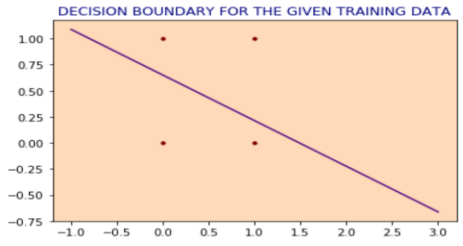
*1.RANDOM INITIAL WEIGHT - [*0.07985933 0.83909476 0.21603111*]*

(FIRST EXAMPLE SHOWS ALL PLOTS FOR REACHING FINAL DECISION BOUNDARY)

```
Before updation w value :  [0.07985933 0.83909476 0.21603111]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.21603111254437346
Error observed :  0
After Updation w value :  [0.07985933 0.83909476 0.21603111]
Before updation w value :  [0.07985933 0.83909476 0.21603111]
Input value :  [ 0  1 -1]
Expected output :  0
Returned output :  0.6230636454089499
Error observed :  -1
After Updation w value :  [0.07985933 0.63909476 0.41603111]
Before updation w value :  [0.07985933 0.63909476 0.41603111]
Input value :  [ 1  0 -1]
Expected output :  0
Returned output :  -0.336171784807526
Error observed :  0
After Updation w value :  [0.07985933 0.63909476 0.41603111]
Before updation w value :  [0.07985933 0.63909476 0.41603111]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  0.30292297314579725
Error observed :  0
After Updation w value :  [0.07985933 0.63909476 0.41603111]
```

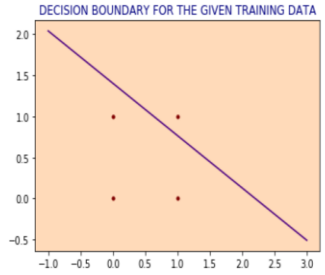



```
updated [0 0 0]

sum of errors 1
Before updation w value :  [0.07985933 0.63909476 0.41603111]
 Before updation w value :  [0.07985933 0.63909476 0.41603111]
 Input value :  [ 0  0 -1]
 Expected output :  0
 Returned output :  -0.41603111254437347
 Error observed :  0
 After Updation w value :  [0.07985933 0.63909476 0.41603111]
 Before updation w value :  [0.07985933 0.63909476 0.41603111]
 Input value :  [ 0  1 -1]
 Expected output :  0
 Returned output :  0.2230636454089498
 Error observed :  -1
 After Updation w value :  [0.07985933 0.43909476 0.61603111]
 Before updation w value :  [0.07985933 0.43909476 0.61603111]
 Input value :  [ 1  0 -1]
 Expected output :  0
 Returned output :  -0.536171784807526
 Error observed :  0
 After Updation w value :  [0.07985933 0.43909476 0.61603111]
 Before updation w value :  [0.07985933 0.43909476 0.61603111]
 Input value :  [ 1  1 -1]
 Expected output :  1
 Returned output :  -0.09707702685420283
 Error observed :  1
 After Updation w value :  [0.27985933 0.63909476 0.41603111]
```

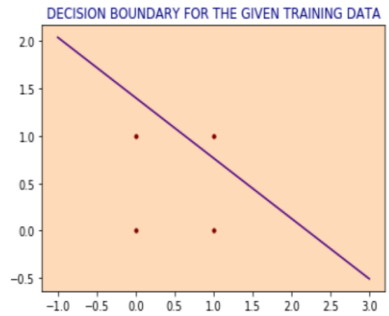



```
 updated [ 1  1 -1]

 sum of errors 2
 Before updation w value :  [0.27985933 0.63909476 0.41603111]
```

```
sum of errors 2
Before updation w value :  [0.27985933 0.63909476 0.41603111]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.41603111254437347
Error observed :  0
After Updation w value :  [0.27985933 0.63909476 0.41603111]
Before updation w value :  [0.27985933 0.63909476 0.41603111]
Input value :  [ 0  1 -1]
Expected output :  0
Returned output :  0.2230636454089498
Error observed :  -1
After Updation w value :  [0.27985933 0.43909476 0.61603111]
Before updation w value :  [0.27985933 0.43909476 0.61603111]
Input value :  [ 1  0 -1]
Expected output :  0
Returned output :  -0.336171784807526
Error observed :  0
After Updation w value :  [0.27985933 0.43909476 0.61603111]
Before updation w value :  [0.27985933 0.43909476 0.61603111]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  0.10292297314579724
Error observed :  0
After Updation w value :  [0.27985933 0.43909476 0.61603111]
```



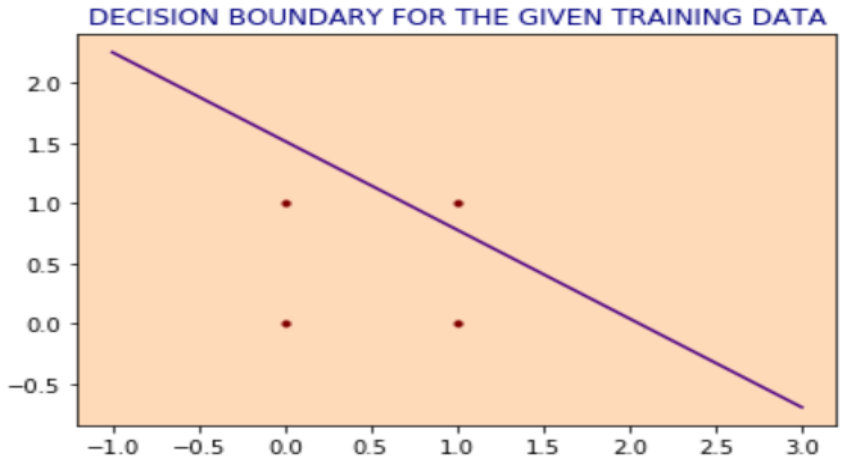

```
updated [0 0 0]
sum of errors 1
Before updation w value :  [0.27985933 0.43909476 0.61603111]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.6160311125443735
Error observed :  0
After Updation w value :  [0.27985933 0.43909476 0.61603111]
Before updation w value :  [0.27985933 0.43909476 0.61603111]
Input value :  [ 0  1 -1]
Expected output :  0
Returned output :  -0.17693635459105023
Error observed :  0
After Updation w value :  [0.27985933 0.43909476 0.61603111]
Before updation w value :  [0.27985933 0.43909476 0.61603111]
Input value :  [ 1  0 -1]
Expected output :  0
Returned output :  -0.336171784807526
Error observed :  0
After Updation w value :  [0.27985933 0.43909476 0.61603111]
Before updation w value :  [0.27985933 0.43909476 0.61603111]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  0.10292297314579724
Error observed :  0
After Updation w value :  [0.27985933 0.43909476 0.61603111]
```




```
BREAKING OFF
```

2.RANDOM INITIAL WEIGHT - [0.35018399 0.87558709 0.12084924*]*

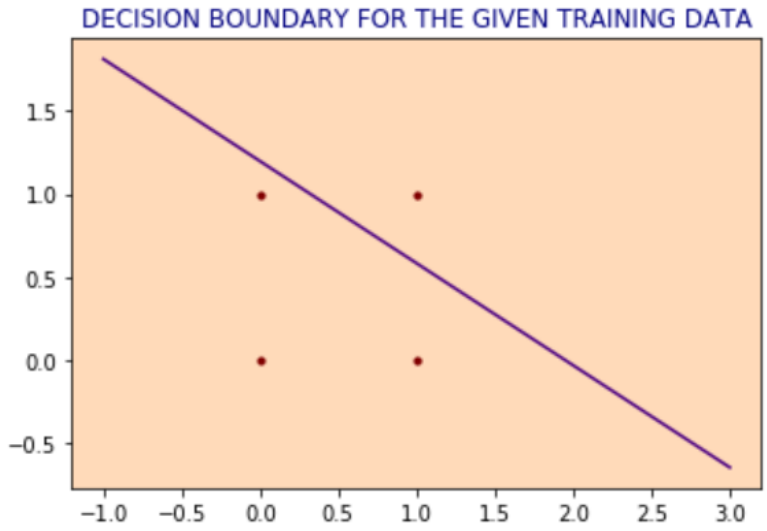
```
updated [0 0 0]

sum of errors 1
Before updation w value :  [0.35018399 0.47558709 0.72084924]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.7208492408397444
Error observed :  0
After Updation w value :  [0.35018399 0.47558709 0.72084924]
Before updation w value :  [0.35018399 0.47558709 0.72084924]
Input value :  [ 0  1 -1]
Expected output :  0
Returned output :  -0.24526215463362583
Error observed :  0
After Updation w value :  [0.35018399 0.47558709 0.72084924]
Before updation w value :  [0.35018399 0.47558709 0.72084924]
Input value :  [ 1  0 -1]
Expected output :  0
Returned output :  -0.37066524647934074
Error observed :  0
After Updation w value :  [0.35018399 0.47558709 0.72084924]
Before updation w value :  [0.35018399 0.47558709 0.72084924]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  0.10492183972677793
Error observed :  0
After Updation w value :  [0.35018399 0.47558709 0.72084924]
```




```
BREAKING OFF
```

*3.RANDOM INITIAL WEIGHT - [*0.46101486 0.75131517 0.89918666*]*

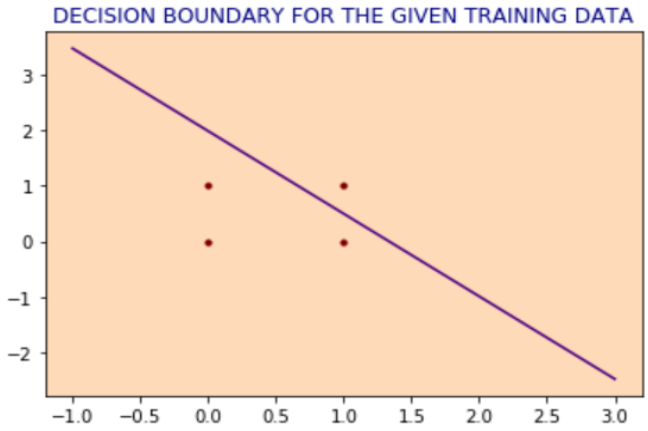
```
Before updation w value :  [0.46101486 0.75131517 0.89918666]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.8991866632732249
Error observed :  0
After Updation w value :  [0.46101486 0.75131517 0.89918666]
Before updation w value :  [0.46101486 0.75131517 0.89918666]
Input value :  [ 0  1 -1]
Expected output :  0
Returned output :  -0.14787149325351312
Error observed :  0
After Updation w value :  [0.46101486 0.75131517 0.89918666]
Before updation w value :  [0.46101486 0.75131517 0.89918666]
Input value :  [ 1  0 -1]
Expected output :  0
Returned output :  -0.4381718034499429
Error observed :  0
After Updation w value :  [0.46101486 0.75131517 0.89918666]
Before updation w value :  [0.46101486 0.75131517 0.89918666]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  0.31314336656976904
Error observed :  0
After Updation w value :  [0.46101486 0.75131517 0.89918666]
```




```
BREAKING OFF
```

*4.RANDOM INITIAL WEIGHT - [*0.54953866 0.16808487 0.73310912*]*

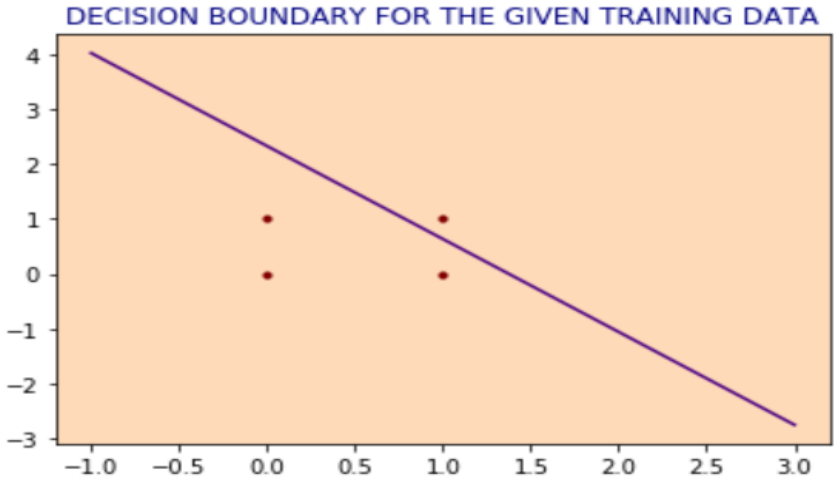
```
sum of errors 1
Before updation w value :  [0.54953866 0.36808487 0.73310912]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.7331091244902537
Error observed :  0
After Updation w value :  [0.54953866 0.36808487 0.73310912]
Before updation w value :  [0.54953866 0.36808487 0.73310912]
Input value :  [ 0  1 -1]
Expected output :  0
Returned output :  -0.3650242561098542
Error observed :  0
After Updation w value :  [0.54953866 0.36808487 0.73310912]
Before updation w value :  [0.54953866 0.36808487 0.73310912]
Input value :  [ 1  0 -1]
Expected output :  0
Returned output :  -0.18357045980137698
Error observed :  0
After Updation w value :  [0.54953866 0.36808487 0.73310912]
Before updation w value :  [0.54953866 0.36808487 0.73310912]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  0.18451440857902246
Error observed :  0
After Updation w value :  [0.54953866 0.36808487 0.73310912]
```




```
BREAKING OFF
```

*5.RANDOM INITIAL WEIGHT - [*0.89846377 0.09439286 0.08507863*]*

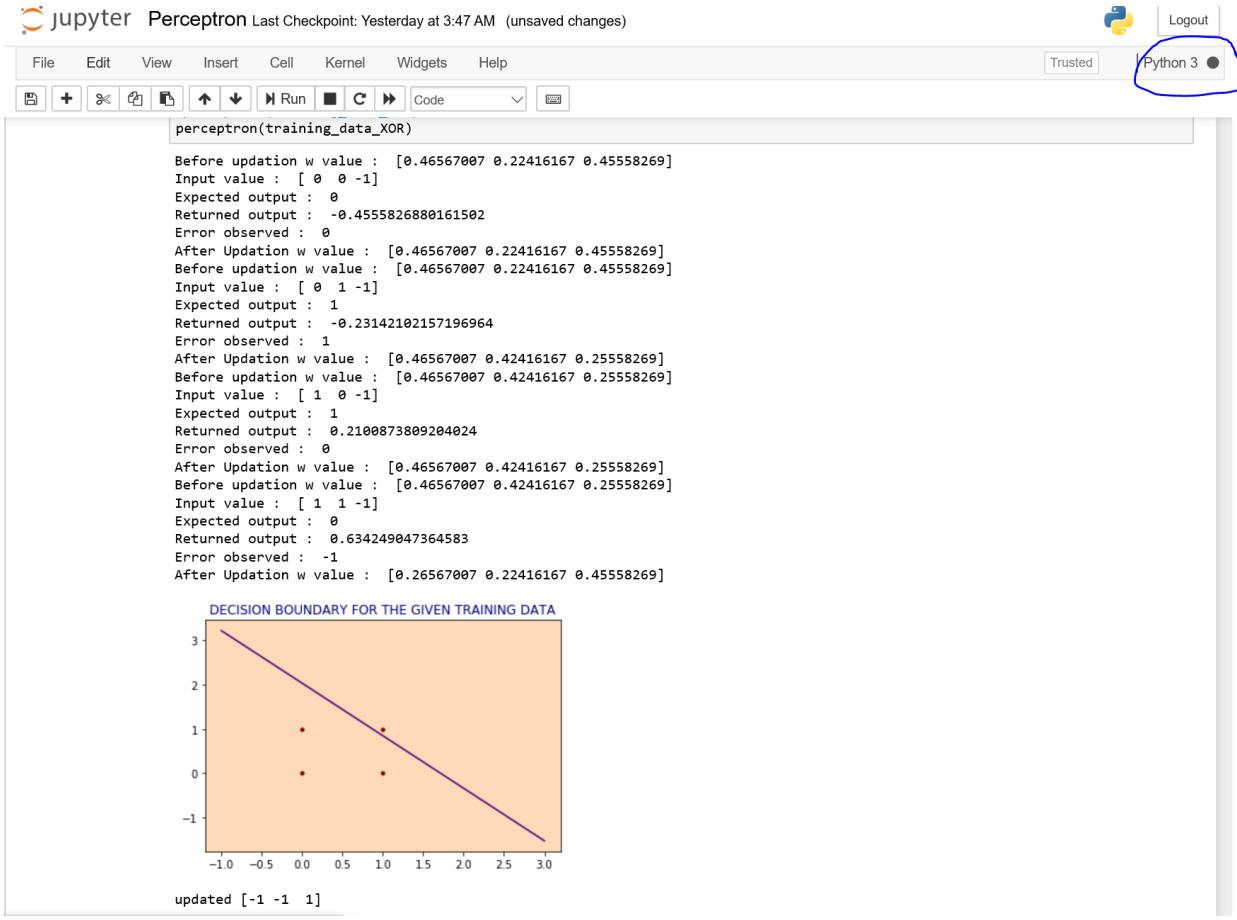
```
sum of errors 1
Before updation w value :  [0.49846377 0.29439286 0.68507863]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.6850786325196472
Error observed :  0
After Updation w value :  [0.49846377 0.29439286 0.68507863]
Before updation w value :  [0.49846377 0.29439286 0.68507863]
Input value :  [ 0  1 -1]
Expected output :  0
Returned output :  -0.39068577507360863
Error observed :  0
After Updation w value :  [0.49846377 0.29439286 0.68507863]
Before updation w value :  [0.49846377 0.29439286 0.68507863]
Input value :  [ 1  0 -1]
Expected output :  0
Returned output :  -0.18661486446344672
Error observed :  0
After Updation w value :  [0.49846377 0.29439286 0.68507863]
Before updation w value :  [0.49846377 0.29439286 0.68507863]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  0.10777799298259183
Error observed :  0
After Updation w value :  [0.49846377 0.29439286 0.68507863]
```




```
BREAKING OFF
```

3. XOR FUNCTION

XOR FUNCTION DOESNOT TERMINATE AND KEEPS RUNNING. SCREENSHOT ATTACHED :

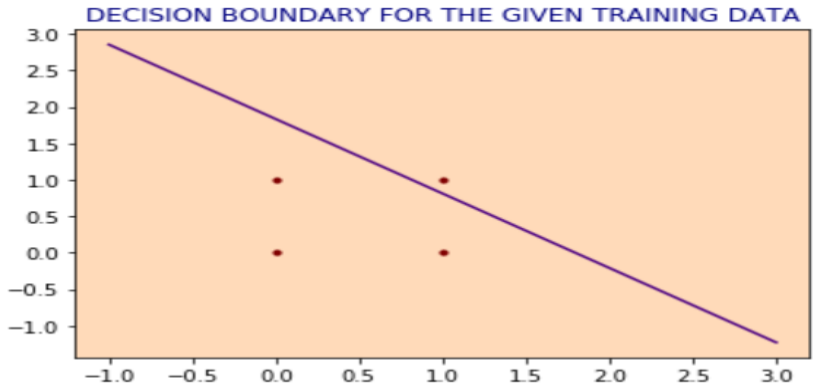


(3) **For the AND and OR problems, is it possible to have the perceptron make zero error without training? Assume the weights are randomly initialized.**

YES, For the AND and OR function, it is possible to have the perceptron make zero error without training for a defined set of input weight values. The chances for that happening is low as the randomly initialized values have to belong to a particular set of w values for the perceptron to make zero error.

For example in case of AND function, if the randomly initialized values happen to become:
= [0.3126865 , 0.30635238, 0.56176929]
Then they can make zero error without training as shown in screenshot:

```
Before updation w value :  [0.3126865, 0.30635238, 0.56176929]
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.56176929
Error observed :  0
After Updation w value :  [0.3126865  0.30635238 0.56176929]
Before updation w value :  [0.3126865  0.30635238 0.56176929]
Input value :  [ 0  1 -1]
Expected output :  0
Returned output :  -0.25541691
Error observed :  0
After Updation w value :  [0.3126865  0.30635238 0.56176929]
Before updation w value :  [0.3126865  0.30635238 0.56176929]
Input value :  [ 1  0 -1]
Expected output :  0
Returned output :  -0.24908279
Error observed :  0
After Updation w value :  [0.3126865  0.30635238 0.56176929]
Before updation w value :  [0.3126865  0.30635238 0.56176929]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  0.0572695899999998
Error observed :  0
After Updation w value :  [0.3126865  0.30635238 0.56176929]
```



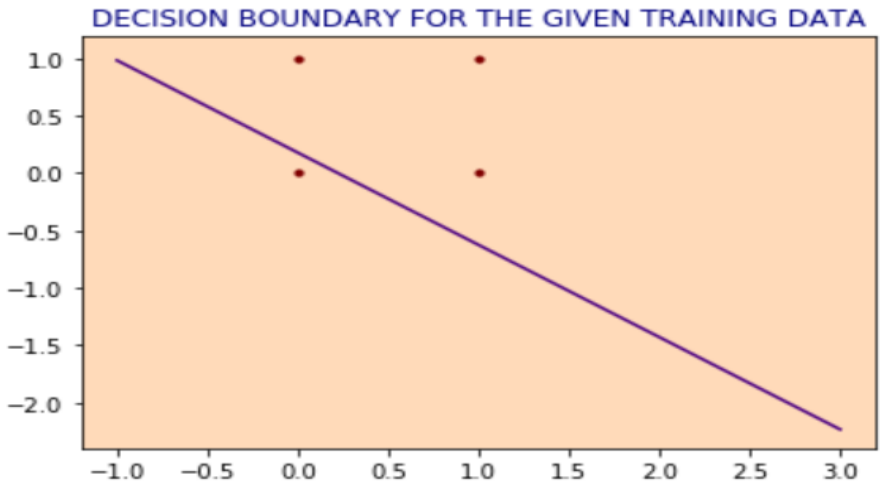

```
BREAKING OFF
```

Similarly for OR function, if the randomly initialized values become :

```
w = [0.65392448, 0.81327696, 0.14541361]
```

Then they can make zero error without training as shown in screenshot:

```
Before updation w value :  [0.65392448, 0.81327696, 0.14541361
Input value :  [ 0  0 -1]
Expected output :  0
Returned output :  -0.14541361
Error observed :  0
After Updation w value :  [0.65392448 0.81327696 0.14541361]
Before updation w value :  [0.65392448 0.81327696 0.14541361]
Input value :  [ 0  1 -1]
Expected output :  1
Returned output :  0.6678633500000001
Error observed :  0
After Updation w value :  [0.65392448 0.81327696 0.14541361]
Before updation w value :  [0.65392448 0.81327696 0.14541361]
Input value :  [ 1  0 -1]
Expected output :  1
Returned output :  0.5085108700000001
Error observed :  0
After Updation w value :  [0.65392448 0.81327696 0.14541361]
Before updation w value :  [0.65392448 0.81327696 0.14541361]
Input value :  [ 1  1 -1]
Expected output :  1
Returned output :  1.32178783
Error observed :  0
After Updation w value :  [0.65392448 0.81327696 0.14541361]
```




```
BREAKING OFF
```

Hence for a particular combination of weight values , the perceptron can make zero error without training.

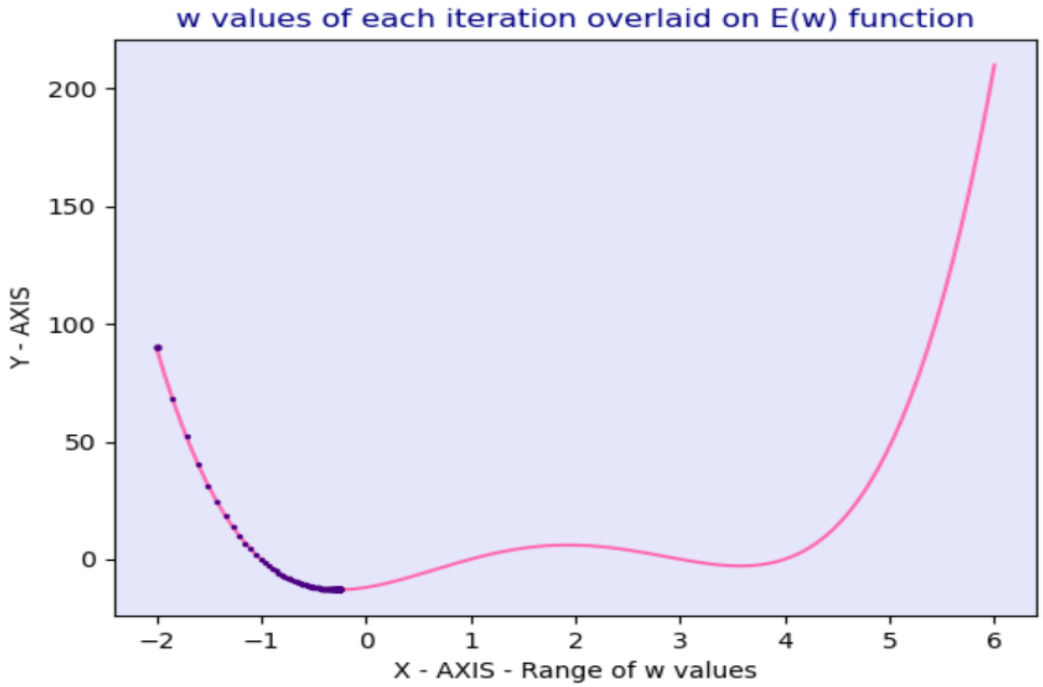
11. **Gradient Descent Programming**

CODE (Python Language)

```
import math
import matplotlib.pyplot as plt
import numpy as np
import sympy as sym
a = sym.Symbol('a')
equation = (a +1)*(a - 1)*(a -3)* (a -4)
Eofw = sym.expand(equation)
print('Resolved Equation : ', Eofw)
print()
dEdw = sym.diff(Eofw)
print("Differentiated equation", dEdw)
print()

range_of_values = [-2 ,-1, 0, 1, 2 ,3 ,4, 5, 6]

w = -2  #Keep varying this value from -2 to 6

eta = 0.001  #Keep varying learning rate
maxerr = 0.0000001
x = np.linspace(-2,6,200)
E = (x+1)*(x-1)*(x-3)*(x-4)
plt.ion()
plt.plot(x,E,color='hotpink',zorder=0)
plt.scatter(w,(w+1)*(w-1)*(w-3)*(w-4),s=5,color='indigo',zorder=5)
plt.show()
while True:
    delta_w = -eta*(4*(w**3)-21*(w**2)+22*w+7)
    updated_w = w + delta_w
    if abs(w-updated_w) < maxerr:
        break
    w=updated_w
    w_vals = plt.scatter(w,(w+1)*(w-1)*(w-3)*(w-4),s=2,color='indigo',zorder=5)
    plt.title("w values of each iteration overlaid on E(w) function", color = 'navy')
    plt.pause(0.01)
    ax = plt.gca()
    ax.set_facecolor('lavender')
    plt.ylabel('Y - AXIS')
    plt.xlabel('X - AXIS - Range of w values')
    print("updated w values : ", w)
print()
print("LOCAL MINIMA REACHED. VALUE IS : ", w)
```

OBSERVATIONS

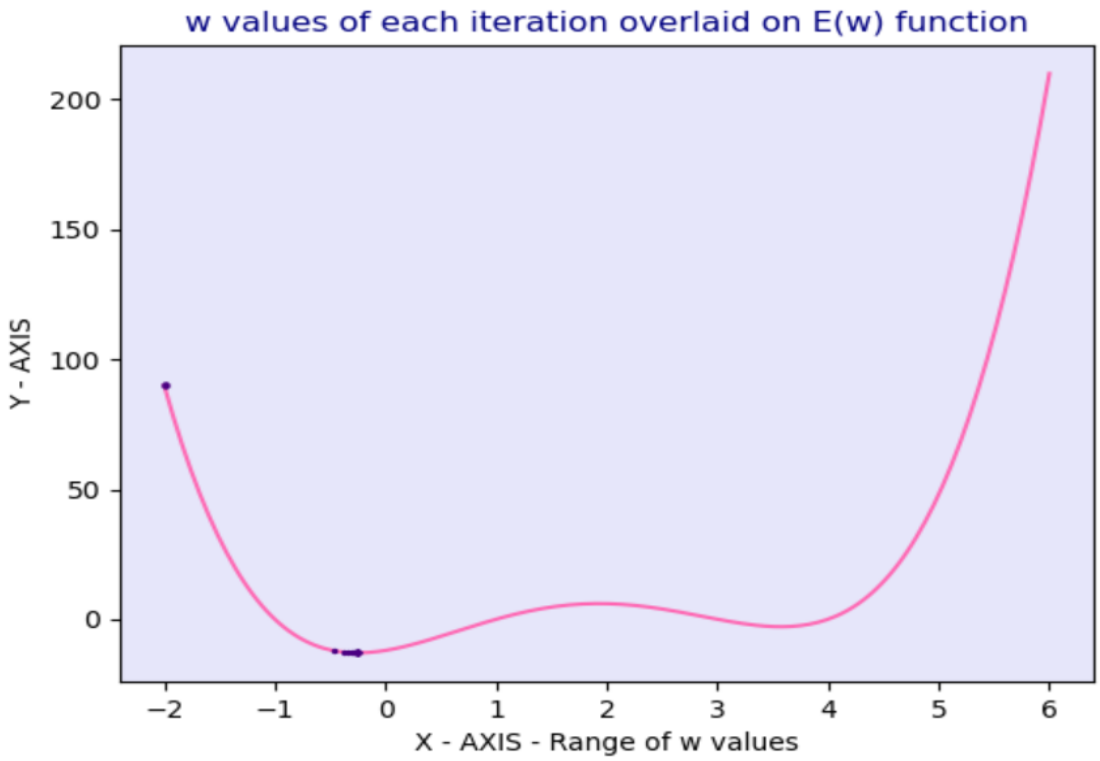
1. **W = -2**
   Learning Rate = 0.001




```
====== RESTART: C:\Users\eunna\OneDrive\Desktop\Unnati\SE
Resolved Equation :  a**4 - 7*a**3 + 11*a**2 + 7*a - 12

Differentiated equation 4*a**3 - 21*a**2 + 22*a + 7

updated w values :   -1.847
updated w values :   -1.716522921308
updated w values :   -1.6036533450165413
updated w values :   -1.5048706991405356
updated w values :   -1.4175742551498969
updated w values :   -1.3397932124431264
updated w values :   -1.270001837880378
updated w values :   -1.2069972318425406
updated w values :   -1.1498159854921561
updated w values :   -1.0976758408439473
updated w values :   -1.049933910600874
updated w values :   -1.0060561533261494
updated w values :   -0.9655946744441034
updated w values :   -0.9281705791202458
updated w values :   -0.8934608351446119
updated w values :   -0.8611880792143062
updated w values :   -0.8311126153938307
updated w values :   -0.8030260679823036
updated w values :   -0.776746298082668
updated w values :   -0.752113296155949
updated w values :   -0.7289858360441817
updated w values :   -0.7072387286889231
updated w values :   -0.6867605522561323
updated w values :   -0.6674517637859894
updated w values :   -0.6492231186833188
updated w values :   -0.6319943403402984
updated w values :   -0.615692943371504
updated w values :   -0.6002535309940313
updated w values :   -0.585616467264515
updated w values :   -0.5717276845884607
updated w values :   -0.5585378237417155
updated w values :   -0.5460017612149571
updated w values :   -0.534078154435405
updated w values :   -0.522729045372261 5
updated w values :   -0.5119195138603787
updated w values :   -0.5016173734290625
updated w values :   -0.4917929036051883
updated w values :   -0.4824186136269092
updated w values :   -0.4734690332990958 4
updated w values :   -0.4649205273778586
updated w values :   -0.4567511304155942 3
updated w values :   -0.4489403994509670 3
updated w values :   -0.4414692823068276
updated w values :   -0.4343199995766902
updated w values :   -0.4274759386478127 5
updated w values :   -0.4209215583348614 5
```

```
updated w values :  -0.25375753804509654
updated w values :  -0.25375726083467315
updated w values :  -0.25375699289153963
updated w values :  -0.2537567339058821
updated w values :  -0.25375648357824426
updated w values :  -0.2537562416191812
updated w values :  -0.2537560077489245
updated w values :  -0.25375578169705876
updated w values :  -0.25375556320220916
updated w values :  -0.2537553520117388
updated w values :  -0.253755147881457
updated w values :  -0.2537549505753365
updated w values :  -0.25375475986524076
updated w values :  -0.2537545755306603
updated w values :  -0.2537543973584573
updated w values :  -0.2537542251426195
updated w values :  -0.253754058684022
updated w values :  -0.25375389779019664
updated w values :  -0.2537537422751098
updated w values :  -0.2537535919589471
updated w values :  -0.2537534466679056
updated w values :  -0.2537533062339928
updated w values :  -0.25375317049483215
updated w values :  -0.2537530392934757
updated w values :  -0.2537529124782223
updated w values :  -0.2537527899024424
updated w values :  -0.25375267142440827
updated w values :  -0.2537525569071304
updated w values :  -0.25375244621819887
updated w values :  -0.2537523392296303
updated w values :  -0.25375223581771994

LOCAL MINIMA REACHED. VALUE IS :  -0.25375223581771994
>>> 
```

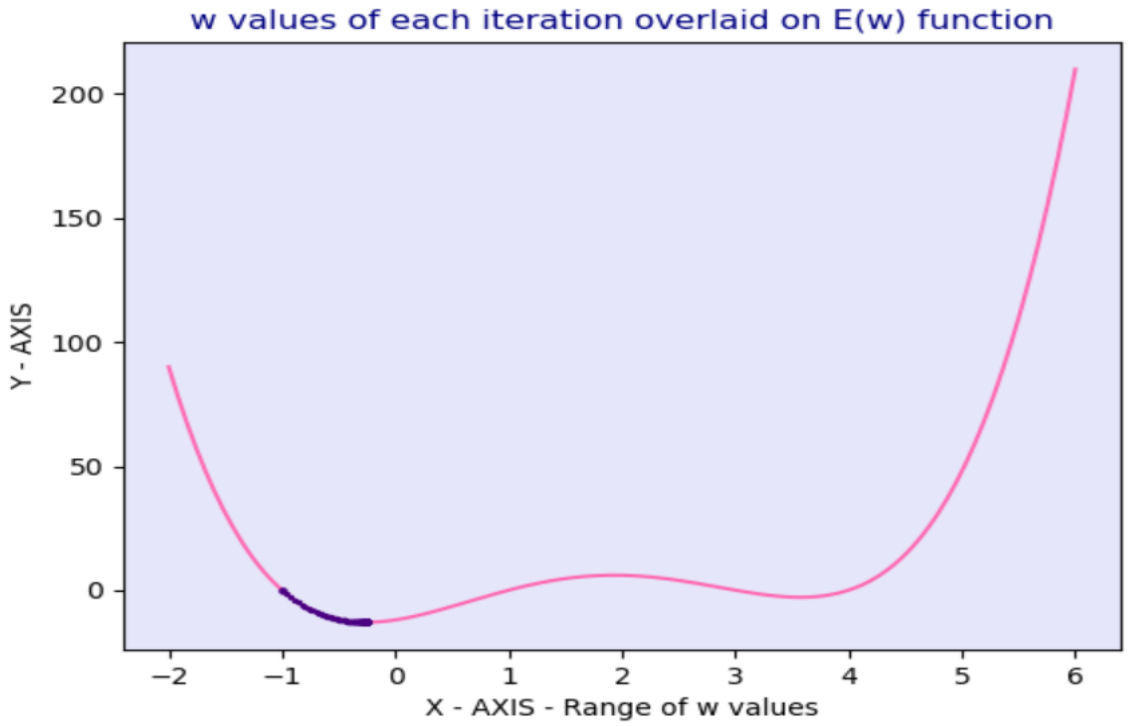
Learning Rate = 0.01



```
Resolved Equation :  a**4 - 7*a**3 + 11*a**2 + 7*a - 12

Differentiated equation 4*a**3 - 21*a**2 + 22*a + 7

updated w values :  -0.47
updated w values :  -0.38605807999999997
updated w values :  -0.33752518892322975
updated w values :  -0.30780768549521864
updated w values :  -0.28902688812389216
updated w values :  -0.276932526632501
updated w values :  -0.2690525934572318
updated w values :  -0.263880209171854
updated w values :  -0.2604686942060568
updated w values :  -0.2582115050364737
updated w values :  -0.25671497454857267
updated w values :  -0.25572141158784184
updated w values :  -0.25506117845123716
updated w values :  -0.2546221837038549
updated w values :  -0.2543301761725561
updated w values :  -0.25413588924531083
updated w values :  -0.25400659790220975
updated w values :  -0.25392054884861615
updated w values :  -0.25386327497720174
updated w values :  -0.2538251517664716
updated w values :  -0.2537997749367189
updated w values :  -0.2537828823862731
updated w values :  -0.25377163737968966
updated w values :  -0.2537641517451979 5
updated w values :  -0.25375916863536674
updated w values :  -0.2537558514155034
updated w values :  -0.2537536431598186
updated w values :  -0.253752173132777
updated w values :  -0.2537511945403119
updated w values :  -0.2537505430937518
updated w values :  -0.25375010942714815
updated w values :  -0.25374982073609
updated w values :  -0.2537496285549349
updated w values :  -0.2537495006202387 7

LOCAL MINIMA REACHED. VALUE IS :  -0.25374950062023877
>>>
```

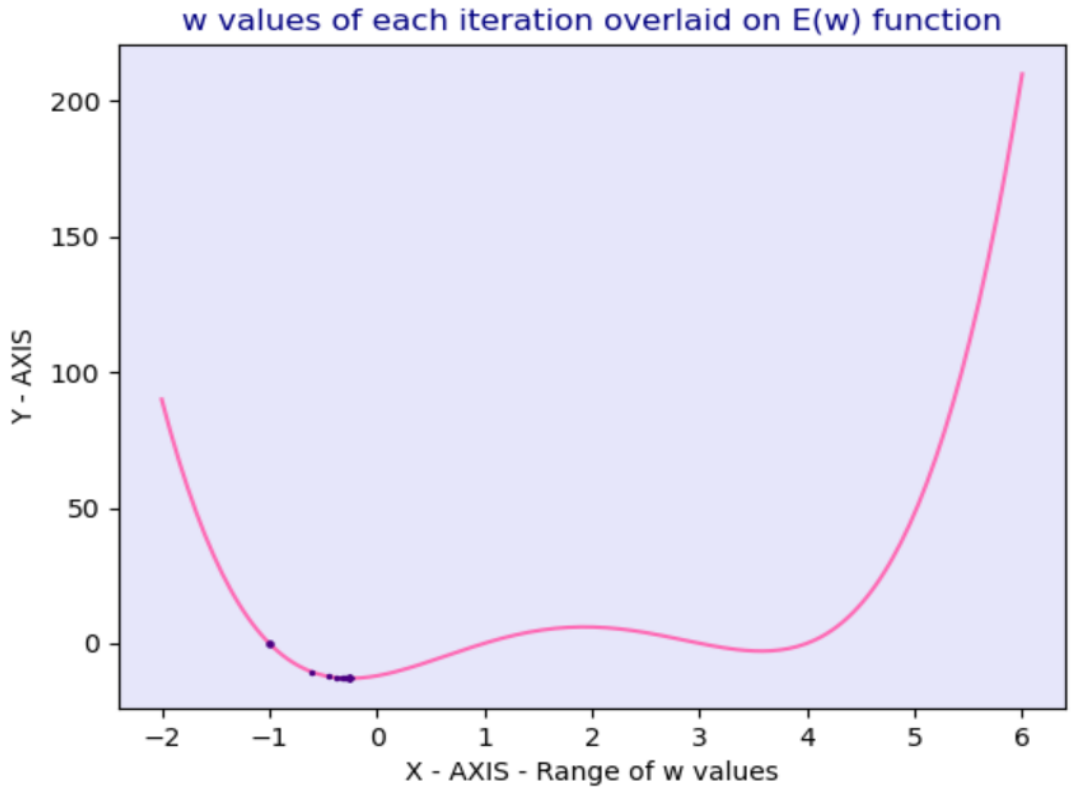
2. W= -1

   <u>Learning Rate = 0.001</u>



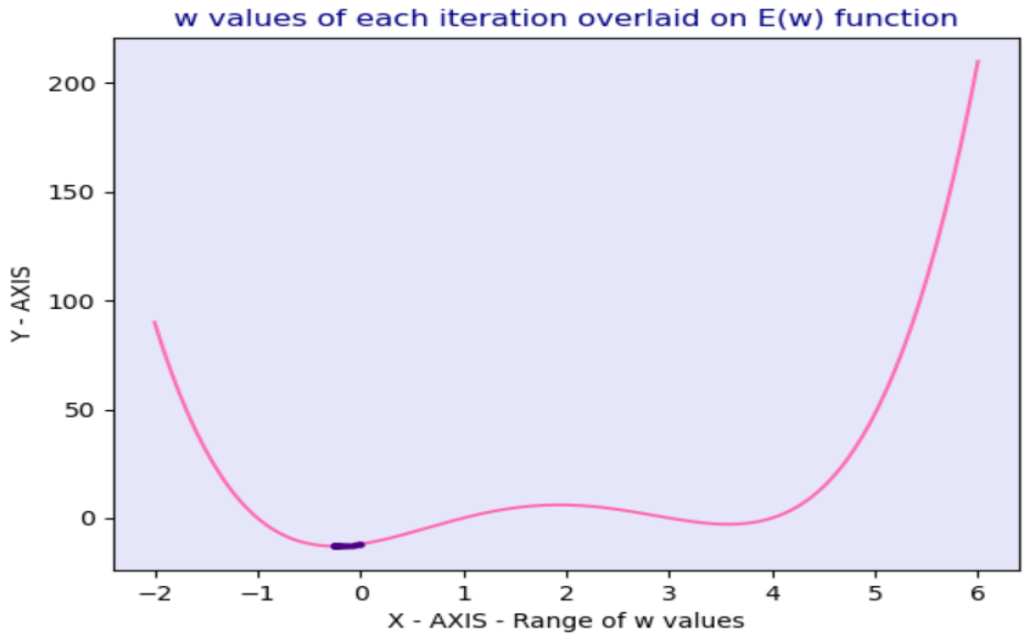
```
updated w values :  -0.25375644805793324
updated w values :  -0.2537562072863312
updated w values :  -0.2537559745638377
updated w values :  -0.2537557496213646
updated w values :  -0.25375553219881974
updated w values :  -0.2537553220448062
updated w values :  -0.2537551189163317
updated w values :  -0.25375492257852733
updated w values :  -0.2537547328043763
updated w values :  -0.25375454937445124
updated w values :  -0.25375437207666046
updated w values :  -0.2537542007060028
updated w values :  -0.2537540350643305
updated w values :  -0.25375387496012
updated w values :  -0.2537537202082507
updated w values :  -0.2537535706297908
updated w values :  -0.2537534260517903
updated w values :  -0.25375328630708105
updated w values :  -0.2537531512340836
updated w values :  -0.25375302067662014
updated w values :  -0.2537528944837341
updated w values :  -0.2537527725095155
updated w values :  -0.2537526546129322
updated w values :  -0.253752540657667
updated w values :  -0.2537524305119597
updated w values :  -0.2537523240484552
updated w values :  -0.25375222211440558

LOCAL MINIMA REACHED. VALUE IS :  -0.25375222211440558
```

Learning Rate = 0.01



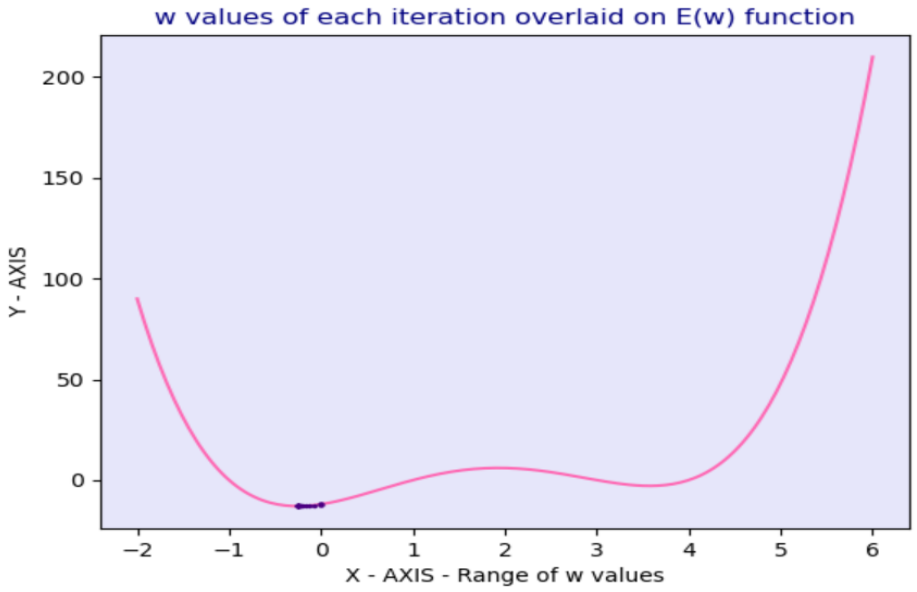
```
Resolved Equation :  a**4 - 7*a**3 + 11*a**2 + 7*a - 12

Differentiated equation 4*a**3 - 21*a**2 + 22*a + 7

updated w values :  -0.6
updated w values :  -0.45376
updated w values :  -0.37695705754730496
updated w values :  -0.33204364100832745
updated w values :  -0.304376562254081
updated w values :  -0.2868302895315236
updated w values :  -0.2755066670401698
updated w values :  -0.2681188948587086
updated w values :  -0.2632653337407387
updated w values :  -0.26006228432137973
updated w values :  -0.2579422341366608
updated w values :  -0.25653628226672764
updated w values :  -0.25560270372645205
updated w values :  -0.2549822640332l606
updated w values :  -0.2545696987860159
updated w values :  -0.25429525840738193
updated w values :  -0.2541126539984078
updated w values :  -0.25399113441999416
updated w values :  -0.25391025668538025
updated w values :  -0.2538564243201986
updated w values :  -0.2538205916575726
updated w values :  -0.25379673943972747
updated w values :  -0.25378086173149356
updated w values :  -0.2537702922641842
updated w values :  -0.25376325631763735
updated w values :  -0.2537585725566518
updated w values :  -0.25375545460945154
updated w values :  -0.25375337900777817
updated w values :  -0.25375199728765546
updated w values :  -0.25375107748068765
updated w values :  -0.2537504651674214
updated w values :  -0.2537500575517404
updated w values :  -0.25374978620272065
updated w values :  -0.2537496055661259
updated w values :  -0.2537494853166232

LOCAL MINIMA REACHED. VALUE IS :  -0.2537494853166232
>>>
```

3. W= 0
   Learning Rate = 0.001



```
updated w values :   -0.2537449366059673
updated w values :   -0.25374508066453033
updated w values :   -0.2537452199072258
updated w values :   -0.253745354495047
updated w values :   -0.2537454845836051
updated w values :   -0.2537456103233094
updated w values :   -0.2537457318595411
updated w values :   -0.25374584933282135
updated w values :   -0.25374596287897383
updated w values :   -0.2537460726292816
updated w values :   -0.2537461787106391
updated w values :   -0.25374628124569865

LOCAL MINIMA REACHED. VALUE IS :  -0.25374628124569865
>>>
```

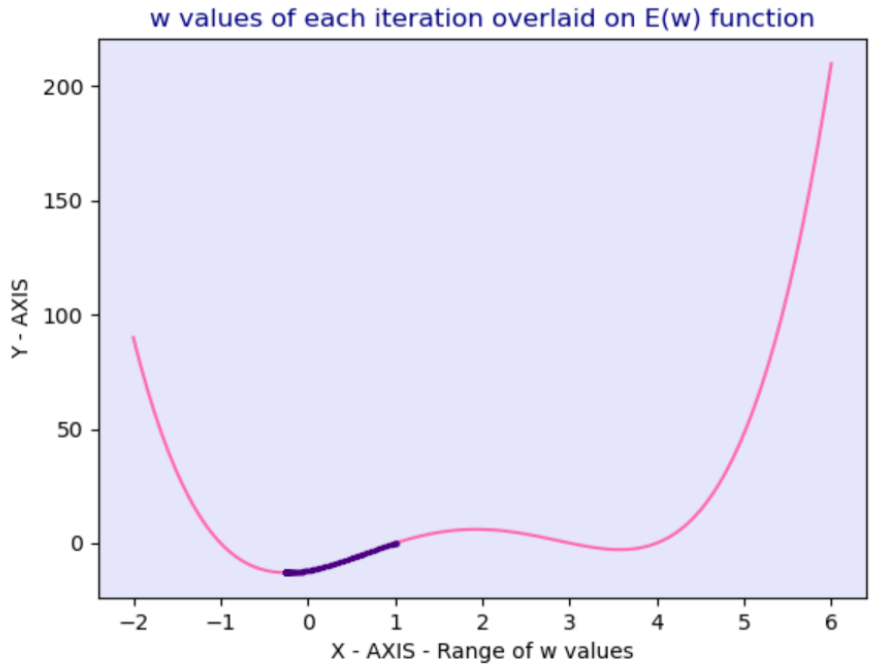
Learning Rate = 0.01




```
Resolved Equation :  a**4 - 7*a**3 + 11*a**2 + 7*a - 12
Differentiated equation 4*a**3 - 21*a**2 + 22*a + 7

updated w values :   -0.07
updated w values :   -0.12355728
updated w values :   -0.16309328309589283
updated w values :   -0.19145335536381963
updated w values :   -0.21135549163762274
updated w values :   -0.22509868360703747
updated w values :   -0.2344801708016593 4
updated w values :   -0.2408328559139121
updated w values :   -0.24511079337016892
updated w values :   -0.24798072220756573
updated w values :   -0.2499011538070202 2
updated w values :   -0.2511840178212878 5
updated w values :   -0.2520399953737465
updated w values :   -0.2526106977929215 6
updated w values :   -0.2529910042867095
updated w values :   -0.2532443472254705
updated w values :   -0.2534130742176883
updated w values :   -0.2535254296717928 7
updated w values :   -0.2536002396762188 6
updated w values :   -0.2536500473030890 4
updated w values :   -0.2536832071467758
updated w values :   -0.2537052829282424 6
updated w values :   -0.2537199793306118 6
updated w values :   -0.2537297629649709 6
updated w values :   -0.253736276032083 2
updated w values :   -0.2537406118230539 6
updated w values :   -0.253743498175881 9
updated w values :   -0.2537454196271166 6
updated w values :   -0.2537466987390813 4
updated w values :   -0.2537475502441818 3
updated w values :   -0.253748117090867 8
updated w values :   -0.2537484944403406 4
updated w values :   -0.2537487456415553 6
updated w values :   -0.253748912865946 8
updated w values :   -0.2537490241870361 5

LOCAL MINIMA REACHED. VALUE IS :  -0.25374902418703615
>>>
```

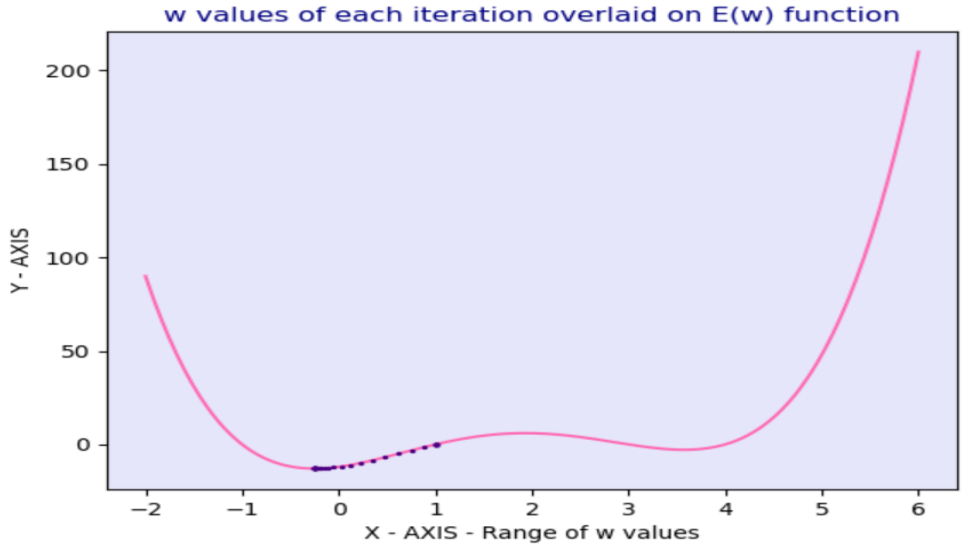
4. W = 1

Learning Rate = 0.001




```
updated w values :  -0.25374530980773113
updated w values :  -0.25374544139018373
updated w values :  -0.253745568573842
updated w values :  -0.2537456915057566
updated w values :  -0.25374581032806237
updated w values :  -0.2537459251781427
updated w values :  -0.25374603618878827
updated w values :  -0.2537461434883507
updated w values :  -0.2537462472008908
updated w values :  -0.25374634744632213

LOCAL MINIMA REACHED. VALUE IS :  -0.25374634744632213
```

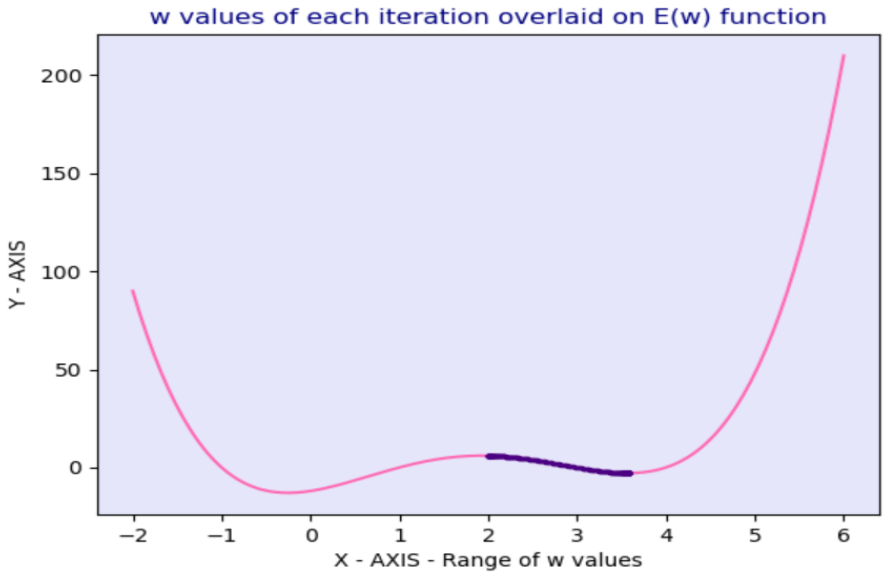
Learning Rate = 0.01



```
------------------------------------------------------------
Resolved Equation :  a**4 - 7*a**3 + 11*a**2 + 7*a - 12

Differentiated equation 4*a**3 - 21*a**2 + 22*a + 7

updated w values :  0.88
updated w values :  0.75176512
updated w values :  0.618064034457854
updated w values :  0.48286651218244236
updated w values :  0.3510960863416672
updated w values :  0.22801016135291435
updated w values :  0.11829138145795398
updated w values :  0.025139566885253956
updated w values :  -0.05025905381269638
updated w values :  -0.10866652963147974
updated w values :  -0.15222879885583146
updated w values :  -0.18373089797974468
updated w values :  -0.20597303294898395
updated w values :  -0.2214002034036219
updated w values :  -0.23196426389050762
updated w values :  -0.23913331175261138
updated w values :  -0.24396819654331392
updated w values :  -0.24721504813407688
updated w values :  -0.24938918406384006
updated w values :  -0.25084219082866305
updated w values :  -0.2518119921257444
updated w values :  -0.2524587165102494
updated w values :  -0.252889741807195
updated w values :  -0.2531768975452077
updated w values :  -0.25336815558616693
updated w values :  -0.2534955196520759
updated w values :  -0.2535803251815052
updated w values :  -0.2536367887280288
updated w values :  -0.25367438025617817
updated w values :  -0.2536994065798122
updated w values :  -0.2537160673207091
updated w values :  -0.2537271586864235
updated w values :  -0.2537345423412734
updated w values :  -0.2537394576958427
updated w values :  -0.253742729869814
updated w values :  -0.2537449081645196
updated w values :  -0.2537463582581041
updated w values :  -0.25374732358600144
updated w values :  -0.2537479662046202
updated w values :  -0.2537483939954781
updated w values :  -0.2537486787755047
updated w values :  -0.2537488683532883
updated w values :  -0.25374899455501077

LOCAL MINIMA REACHED. VALUE IS :  -0.25374899455501077
```

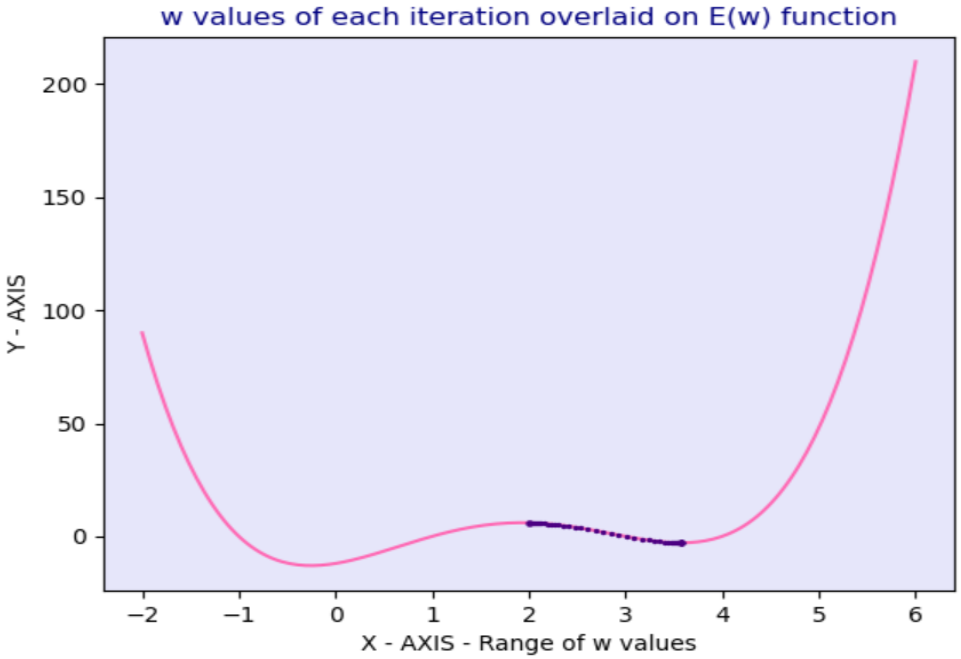
5. W = 2

   Learning Rate = 0.001



```
updated w values :  3.5742085798845595
updated w values :  3.5742087113063303
updated w values :  3.574208839418527
updated w values :  3.574208964304493
updated w values :  3.574209086045474
updated w values :  3.5742092047206686
updated w values :  3.5742093204072813
updated w values :  3.5742094331805725
updated w values :  3.574209543113908
updated w values :  3.5742096502788048
updated w values :  3.57420975474498
updated w values :  3.574209856580395

LOCAL MINIMA REACHED. VALUE IS :  3.574209856580395
>>>
```

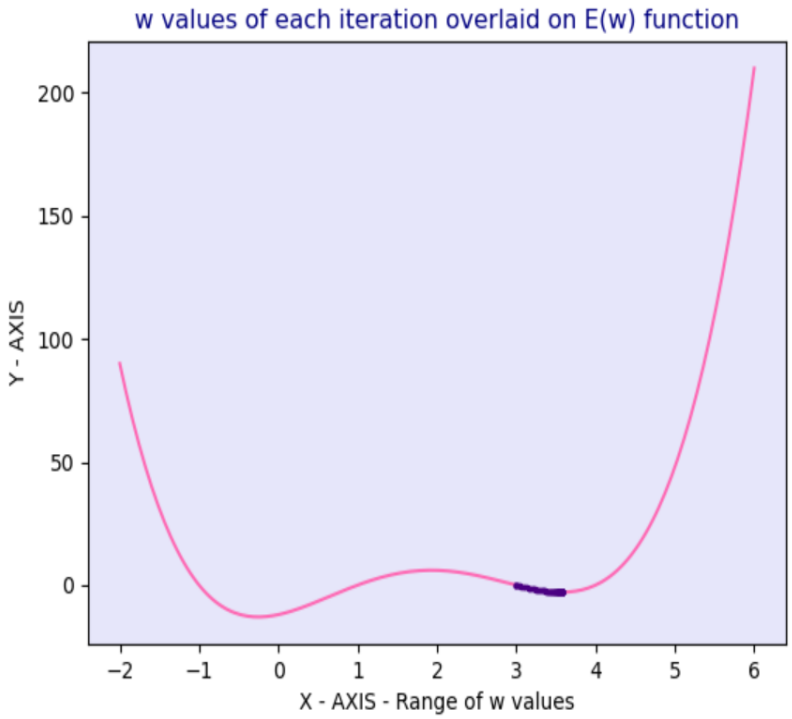
Learning Rate = 0.01




```
updated w values :  3.574212279291063
updated w values :  3.574212661890033
updated w values :  3.5742129481390523
updated w values :  3.5742131623019
updated w values :  3.5742133225320334
updated w values :  3.5742134424113363

LOCAL MINIMA REACHED. VALUE IS :  3.5742134424113363
```

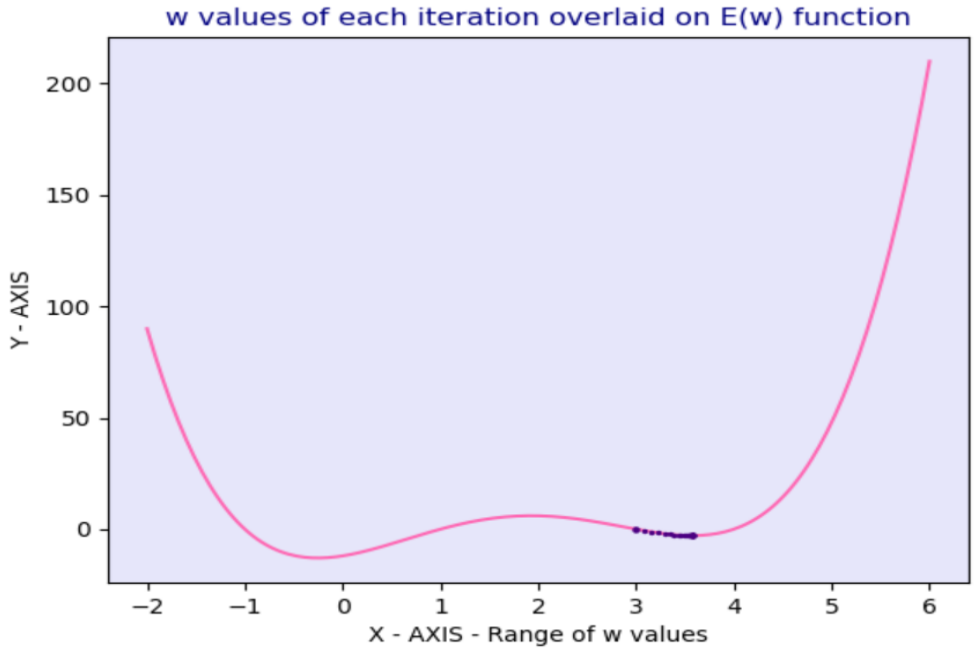
6. W = 3

   Learning Rate = 0.001




```
updated w values :  3.5742094913484563
updated w values :  3.5742095998169567
updated w values :  3.574209705553907
updated w values :  3.5742098086280953
updated w values :  3.5742099091065773

LOCAL MINIMA REACHED. VALUE IS :  3.5742099091065773
>>>
```

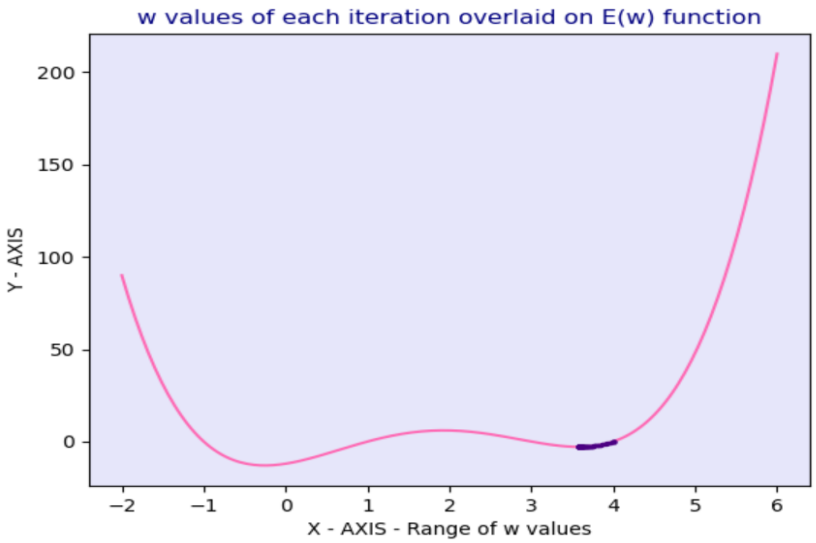
Learning Rate = 0.01



```
updated w values :  3.5742122267972247
updated w values :  3.57421262261572
updated w values :  3.5742129187551974
updated w values :  3.5742131403177915
updated w values :  3.5742133060841925
updated w values :  3.574213430105564

LOCAL MINIMA REACHED. VALUE IS :  3.574213430105564
```

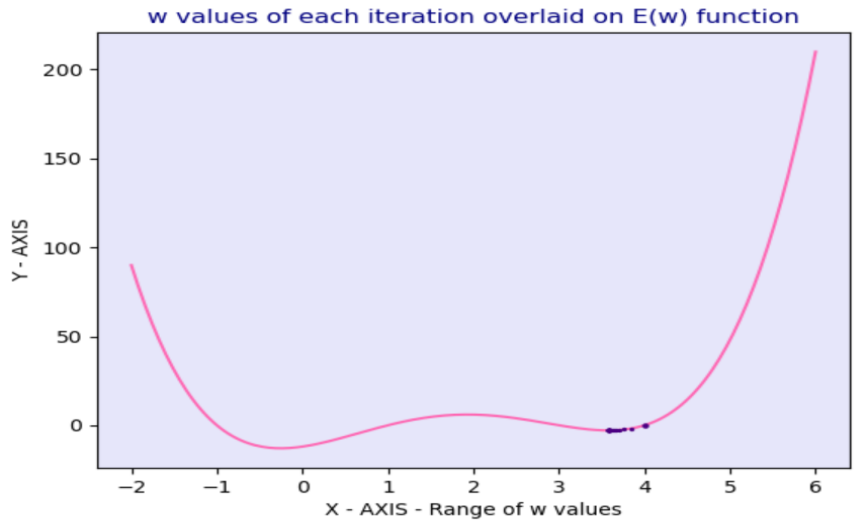
7. W = 4

Learning Rate = 0.001




```
updated w values :  3.5742180937244665
updated w values :  3.5742179855587164
updated w values :  3.574217880116932
updated w values :  3.574217777330515
updated w values :  3.574217677132594

LOCAL MINIMA REACHED. VALUE IS :  3.574217677132594
```

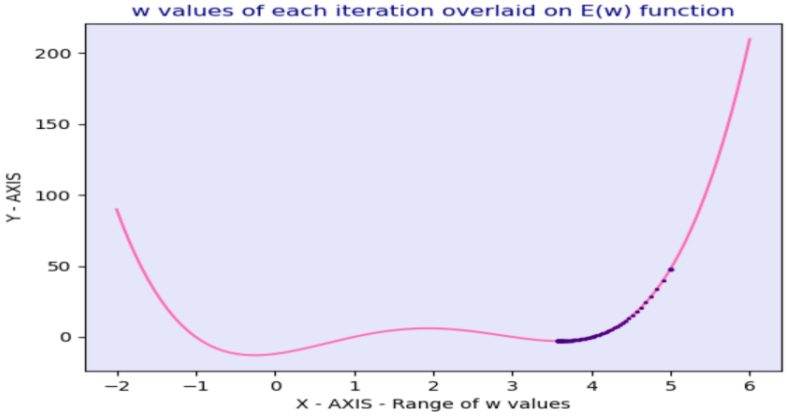
Learning Rate = 0.01



```
Resolved Equation :  a**4 - 7*a**3 + 11*a**2 + 7*a - 12

Differentiated equation 4*a**3 - 21*a**2 + 22*a + 7

updated w values :  3.85
updated w values :  3.7630600000000003
updated w values :  3.7074265256335757
updated w values :  3.6699002922658175
updated w values :  3.643764170595468
updated w values :  3.625176890846107
updated w values :  3.6117689743796513
updated w values :  3.6020005661744214
updated w values :  3.5948331283342077
updated w values :  3.589547127837209
updated w values :  3.5856341131916563
updated w values :  3.582729517119895
updated w values :  3.5805690984178065
updated w values :  3.57895978686158
updated w values :  3.577759666225471
updated w values :  3.576863953703454
updated w values :  3.57619502504242
updated w values :  3.5756952317835013
updated w values :  3.5753216808420114
updated w values :  3.5750424133125405
updated w values :  3.574833592341491
updated w values :  3.57467742512852
updated w values :  3.5745606226601323
updated w values :  3.5744732553657204
updated w values :  3.5744079014359955
updated w values :  3.5743590121225015
updated w values :  3.5743224382715084
updated w values :  3.5742950768721458
updated w values :  3.5742746070511853
updated w values :  3.5742592927966306
updated w values :  3.5742478354981677
updated w values :  3.574239263698874
updated w values :  3.574232850653602
updated w values :  3.574228052672621
updated w values :  3.574224463005617
updated w values :  3.574221777346613
updated w values :  3.5742197680300047
updated w values :  3.5742182647271754
updated w values :  3.574217140005611
updated w values :  3.5742162985254082
updated w values :  3.5742156689568443
updated w values :  3.5742151979335866
updated w values :  3.574214845528791
updated w values :  3.5742145818705366
updated w values :  3.5742143846096357
updated w values :  3.574214237025148
updated w values :  3.5742141266070013

LOCAL MINIMA REACHED. VALUE IS :  3.5742141266070013
```

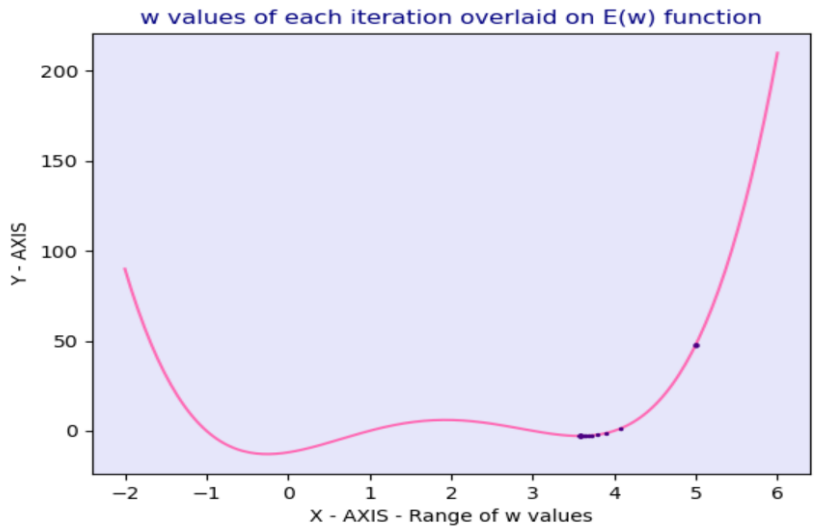
8. W = 5

   Learning Rate = 0.001



```
updated w values :  3.5742179798852476
updated w values :  3.5742178745863398
updated w values :  3.574217771939201
updated w values :  3.5742176718770504

LOCAL MINIMA REACHED. VALUE IS :  3.5742176718770504
>>> |
```

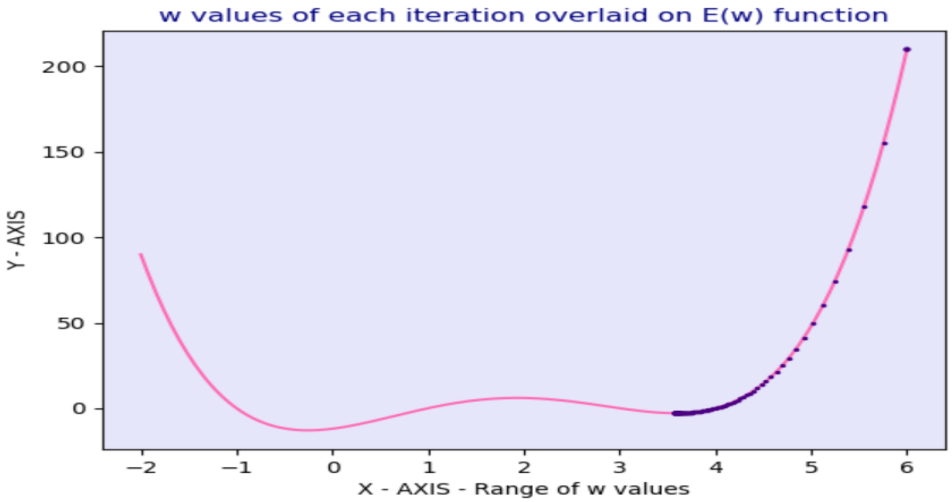
Learning Rate = 0.01




```
updated w values :  3.574215343793884
updated w values :  3.5742149546568913
updated w values :  3.574214663516777
updated w values :  3.5742144456948157
updated w values :  3.5742142827271866
updated w values :  3.574214160799853

LOCAL MINIMA REACHED. VALUE IS :  3.574214160799853
>>> |
```

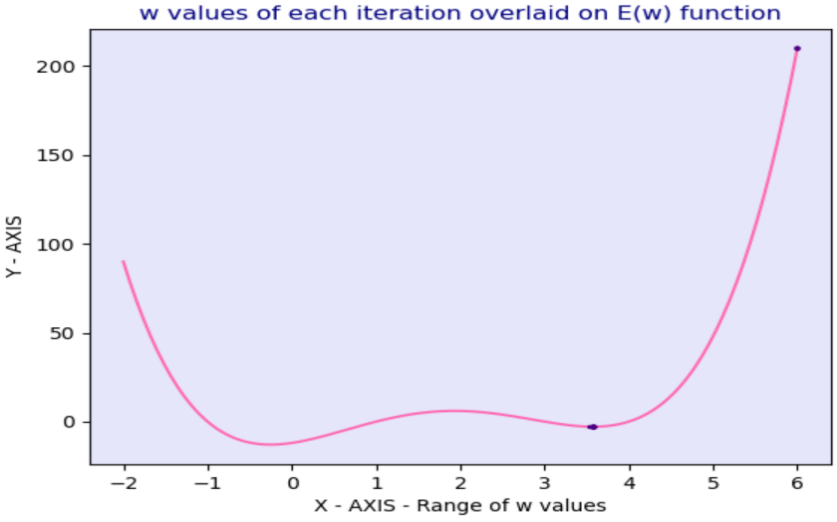
9. W = 6

Learning Rate = 0.001



```
updated w values :  3.574218331999476
updated w values :  3.574218217833183
updated w values :  3.5742181065419705
updated w values :  3.5742179980534337
updated w values :  3.574217892296992
updated w values :  3.5742177892038414
updated w values :  3.574217688706911

LOCAL MINIMA REACHED. VALUE IS :  3.574217688706911
```

Learning Rate = 0.01




```
Resolved Equation :  a**4 - 7*a**3 + 11*a**2 + 7*a - 12

Differentiated equation 4*a**3 - 21*a**2 + 22*a + 7

updated w values :  3.53
updated w values :  3.54070992
updated w values :  3.5489030063687683
updated w values :  3.5551374510514635
updated w values :  3.5598620777138383
updated w values :  3.5634313117092384
updated w values :  3.566121272844388
updated w values :  3.5681449046985914
updated w values :  3.569665184921082
updated w values :  3.5708061401916438
updated w values :  3.571661752866597
updated w values :  3.572303011312736
updated w values :  3.572783407268828
updated w values :  3.5731431759673904
updated w values :  3.5734125407566 8
updated w values :  3.5736141815655023
updated w values :  3.5737651048477717
updated w values :  3.5738780556405496
updated w values :  3.5739625813476312
updated w values :  3.574025831744949
updated w values :  3.5740731598298163
updated w values :  3.574108572653418
updated w values :  3.5741350693461222
updated w values :  3.5741548944246433
updated w values :  3.574169727536 7
updated w values :  3.5741808255498597
updated w values :  3.574189128929567
updated w values :  3.5741953413679 75
updated w values :  3.574199989382208
updated w values :  3.57420346691651
updated w values :  3.574206068719229
updated w values :  3.5742080153183884
updated w values :  3.5742094717098425
updated w values :  3.574210561340414
updated w values :  3.5742113765703407
updated w values :  3.5742119865013398
updated w values :  3.574212442833543
updated w values :  3.5742127842475955
updated w values :  3.574213039683279
updated w values :  3.5742132307925107
updated w values :  3.5742133737746227
updated w values :  3.574213480749478

LOCAL MINIMA REACHED. VALUE IS :  3.574213480749478
>>>
```

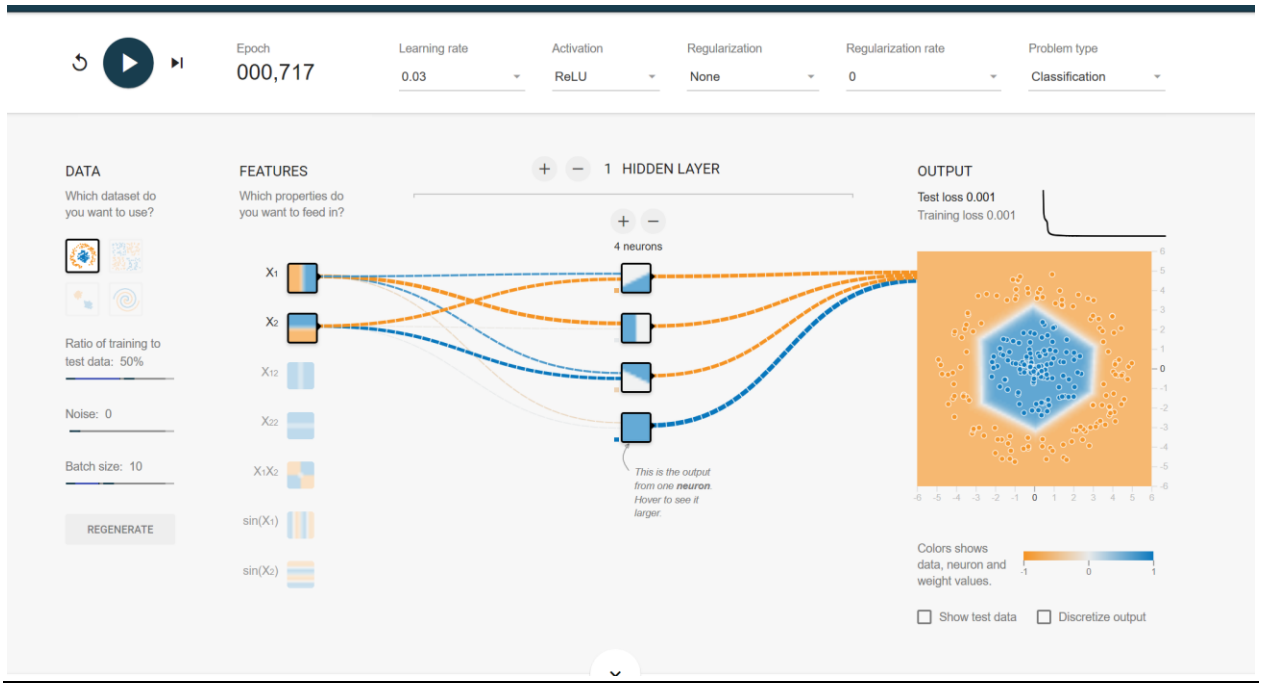
THUS 2 VALUES OF LOCAL MINIMA ARE FOUND : 3.574213480 AND -0.253746

12. PLAYGROUND TENSORFLOW

CIRCLE DATASET

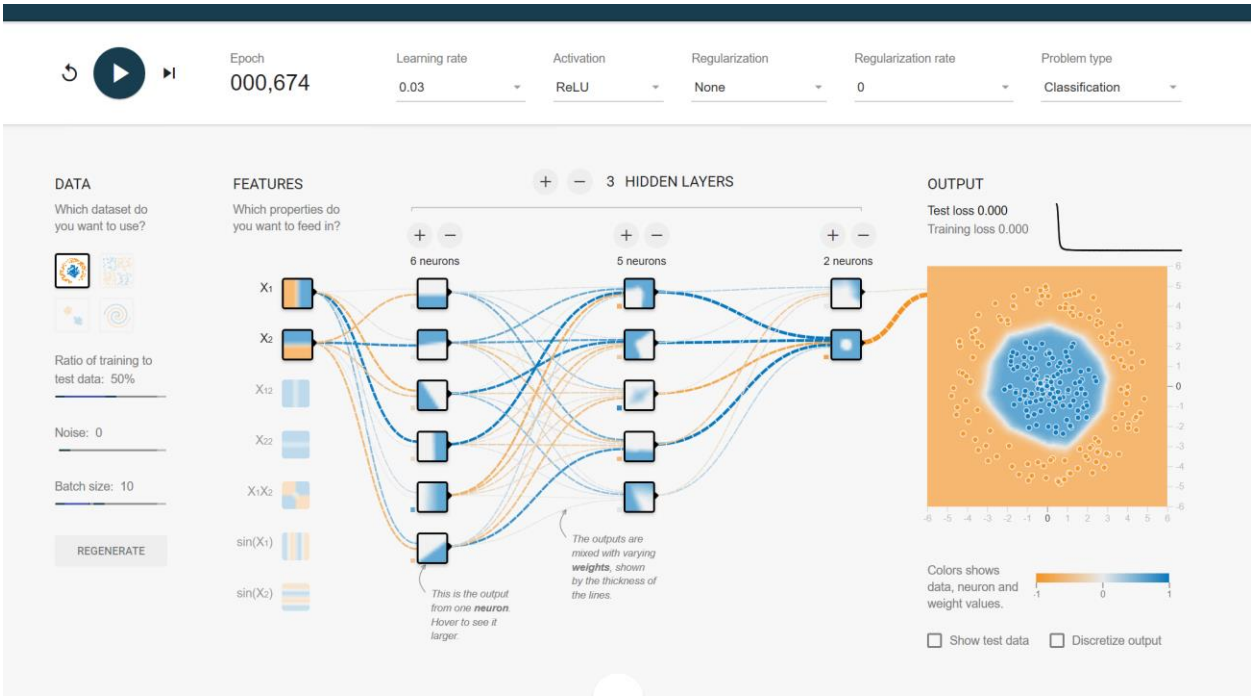
1.

Test Loss – 0.001
Training Loss – 0.001
Number of Hidden Layers – 1
Number of Neurons – 4
Epoch – 717
Input feature – X1, X2



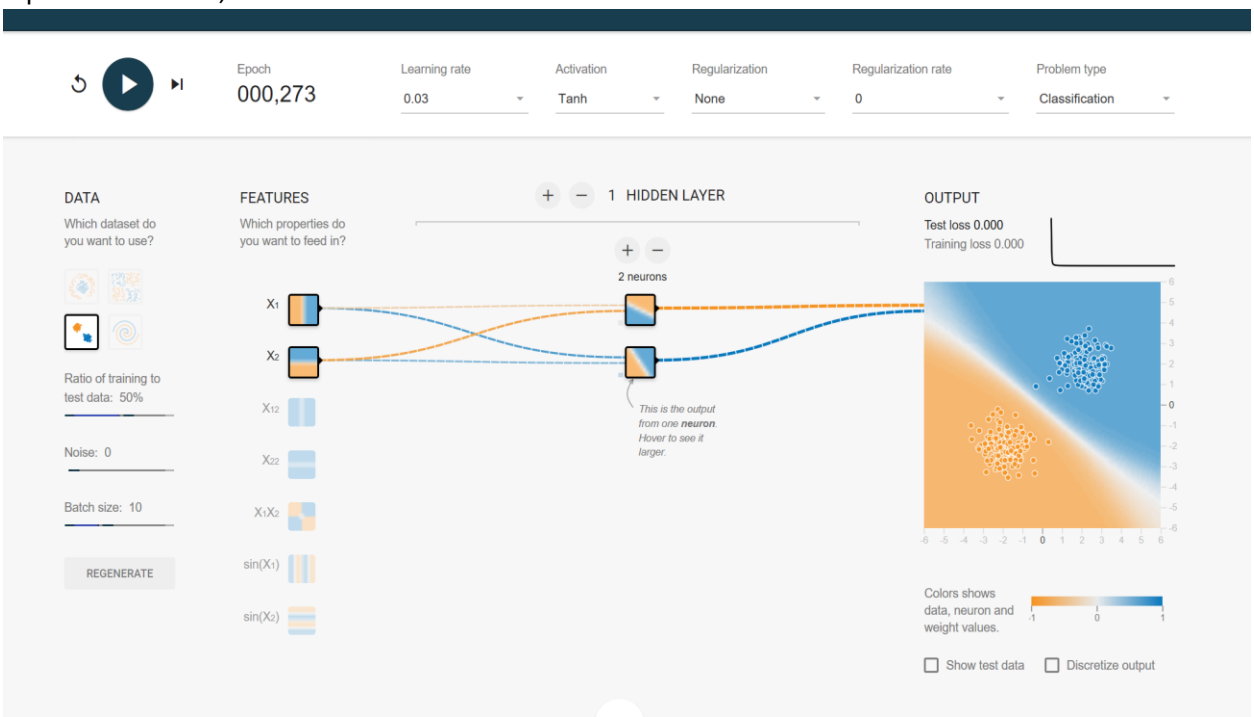
2.

Test Loss – 0.000
Training Loss – 0.000
Number of Hidden Layers – 3
Number of Neurons – 6(1st Layer) , 5(2nd Layer) , 2(3rd Layer)
Epoch – 674
Input feature – X1, X2



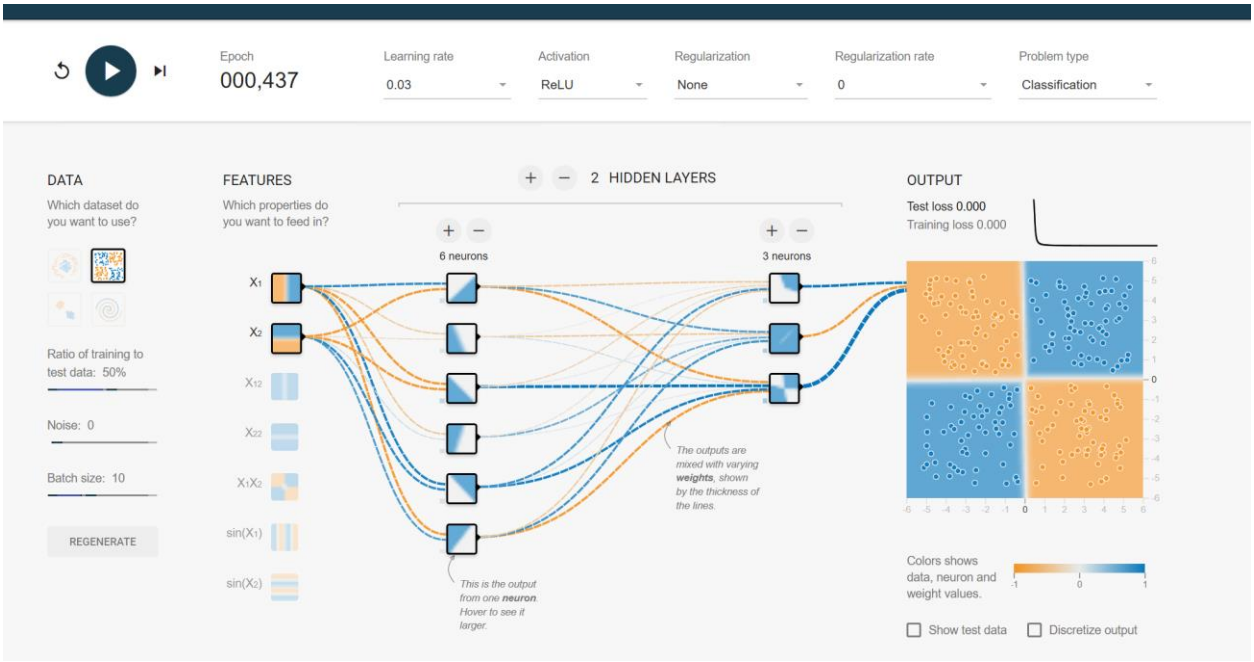
GAUSSIAN

Test Loss – 0.000
Training Loss – 0.000
Number of Hidden Layers – 1
Number of Neurons – 2(1st Layer)
Epoch – 273
Input feature – X1, X2



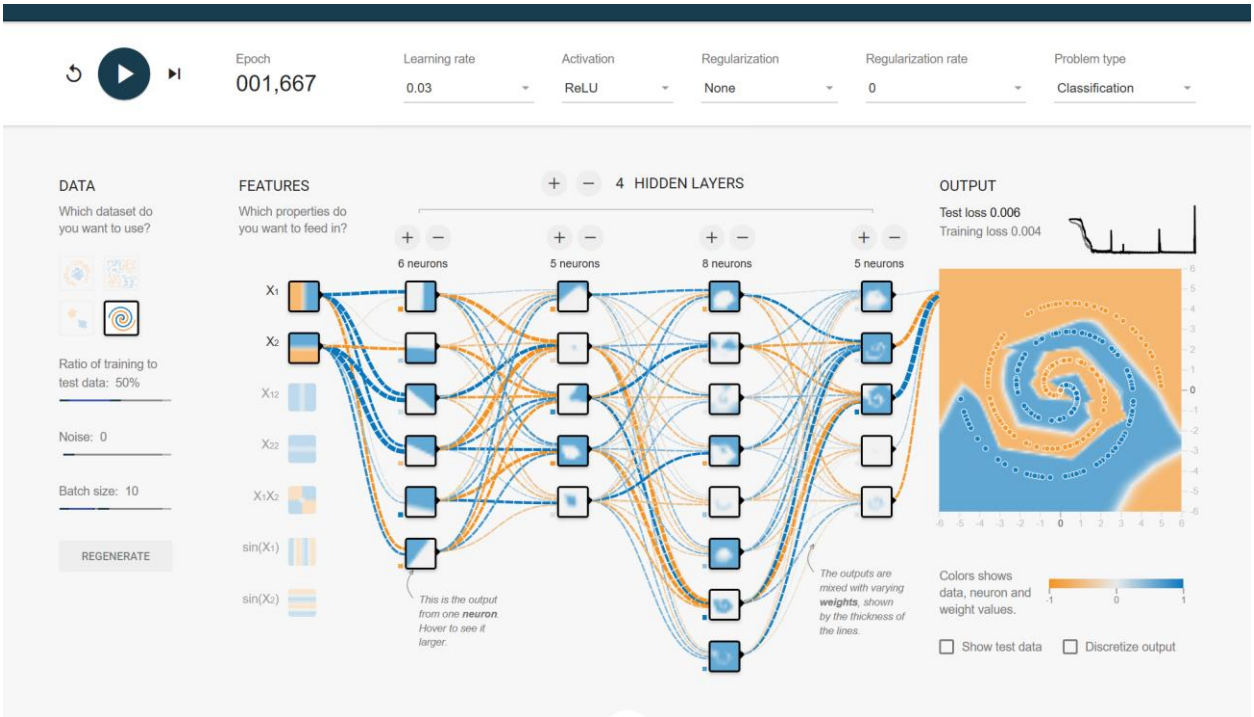
XOR

Test Loss – 0.000
Training Loss – 0.000
Number of Hidden Layers – 2
Number of Neurons – 6(1st Layer) , 3(2nd Layer)
Epoch – 437
Input feature – X1, X2



SPIRAL

1.
Test Loss – 0.006
Training Loss – 0.004
Number of Hidden Layers – 4
Number of Neurons – 6(1st Layer) , 5(2nd Layer), 8(3rd layer), 5(4th layer)
Epoch – 1667
Input feature – **X1, X2**



2.

Test Loss – 0.007

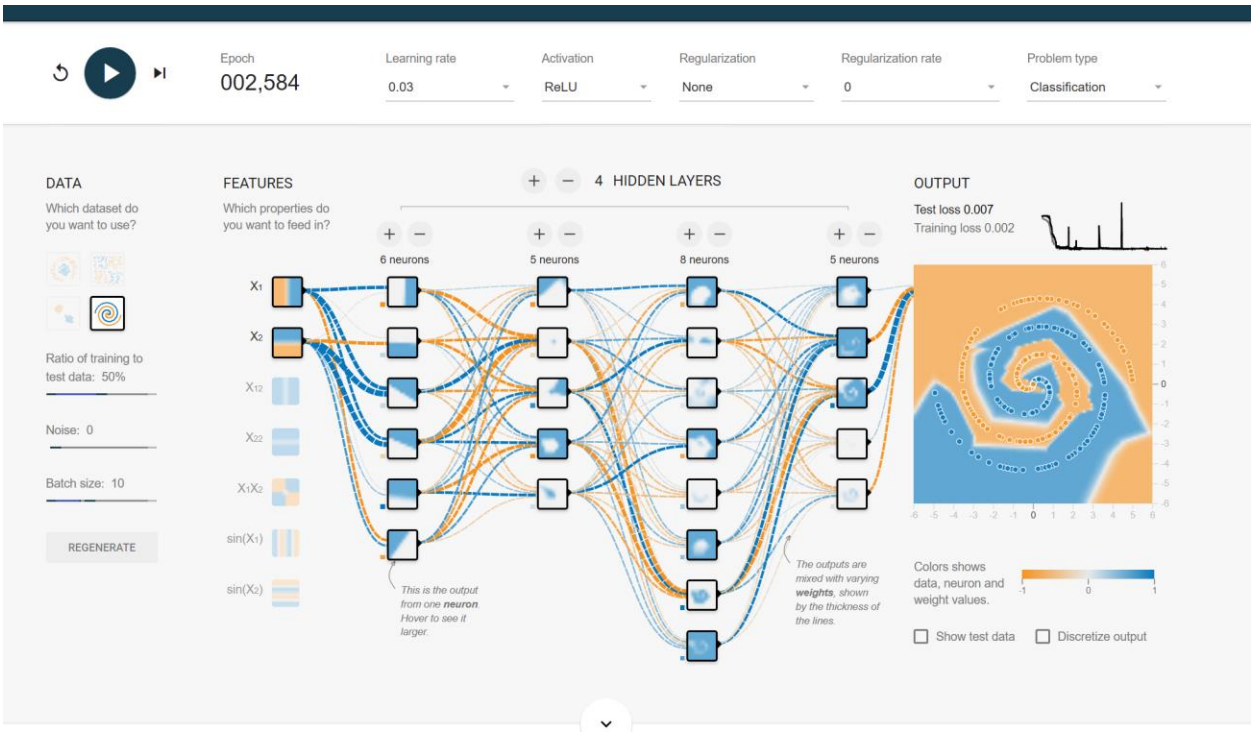
Training Loss – 0.002

Number of Hidden Layers – 4

Number of Neurons – 6(1st Layer) , 5(2nd Layer), 8(3rd layer), 5(4th layer)

Epoch – 2584

Input feature **– X1, X2**



3.

Test Loss – 0.010

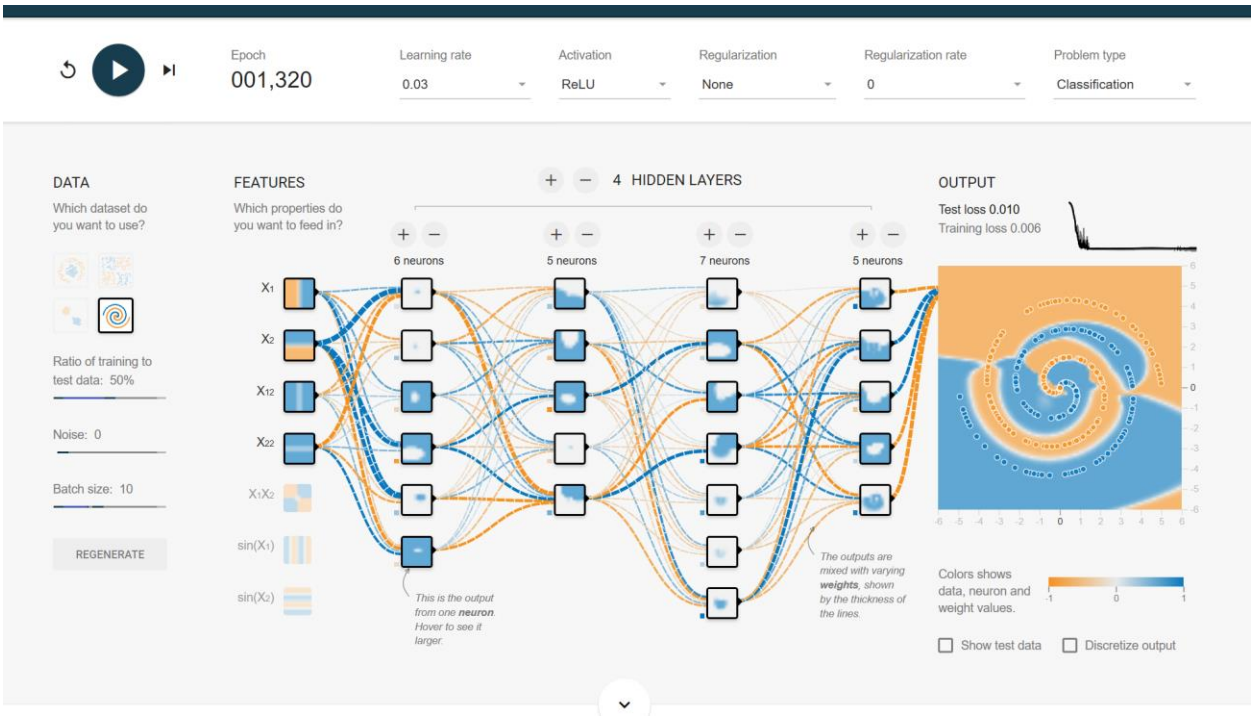
Training Loss – 0.006

Number of Hidden Layers – 4

Number of Neurons – 6(1st Layer) , 5(2nd Layer), 7(3rd layer), 5(4th layer)

Epoch – 1320

Input feature – **X1, X2, X12, X22**



4.

Test Loss – 0.014
Training Loss – 0.005
Number of Hidden Layers – 4
Number of Neurons – 6(1st Layer) , 5(2nd Layer), 8(3rd layer), 5(4th layer)
Epoch – 430
Input feature – **X1, X2, X12, X22**

5.

Test Loss – 0.017

Training Loss – 0.004

Number of Hidden Layers – 4

Number of Neurons – 8(1st Layer) , 7(2nd Layer), 8(3rd layer), 5(4th layer)

Epoch – 693

Input feature – **X1, X2, X12, X22**