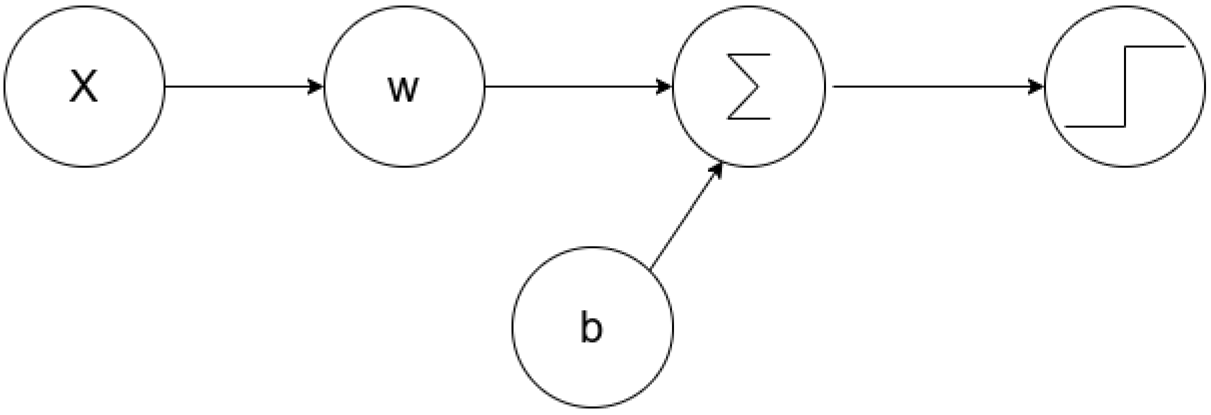
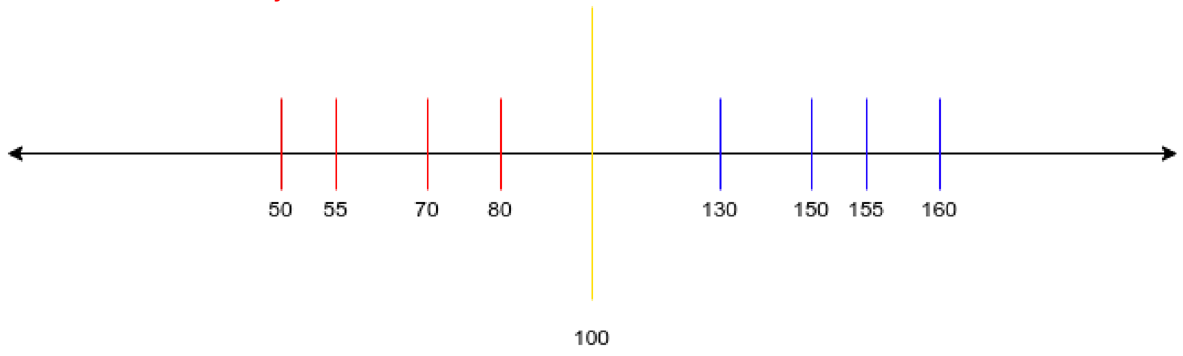
Sheet 2

Q1) Given a data set of gray levels, {50, 55, 70, 80, 130, 150, 155, 160} and their corresponding labels {O, O, O, O, B, B, B, and B}, find and visualize a classification boundary assuming the use of a bipolar perceptron with bias. The symbol O stands for object while B stands for background.

As the input data is one dimensional, we will be using a one input one perceptron model like this:



And after plotting the provided data we were able to make a good assumption about the classification boundary:



Based on the drawing we assumed W = -1 and b = 100 and then computed the NN function as follows:

1. Sign(-1*50 + 100) = 1 …… Correct
2. Sign(-1*55 + 100) = 1 …… Correct
3. Sign(-1*70 + 100) = 1 …… Correct
4. Sign(-1*80 + 100) = 1 …… Correct
5. Sign(-1*130 + 100) = -1 …… Correct
6. Sign(-1*150 + 100) = -1 …… Correct
7. Sign(-1*155 + 100) = -1 …… Correct
8. Sign(-1*160 + 100) = -1 …… Correct

With all the examples computed correctly we have successfully designed a 1D Classifier.

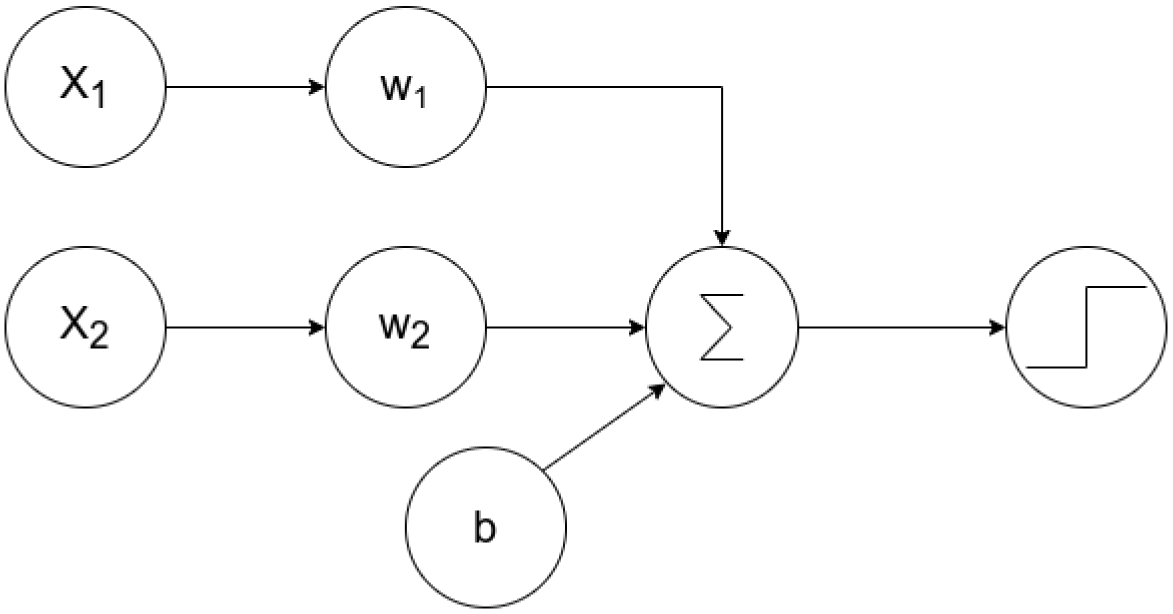
Q2) For the problem description in (1), write a computer program to generate the weighting coefficients randomly and count the misclassified points in each case. The random numbers generated should be between -1 and 1. Use a loop which stops when you reach zero error. How many trials do you get? Comment on the results.

- Step 0: Set $error = 1$ & $trial = 0$;
- Step 1: Select W & B from a random distribution [-1, 1]
- Step 2: while(error) do $error = sum(sign(w * X + b) - labels)$; $trial++$;
- Step 3: if $error > 0$ go to step 0; otherwise print number of trials

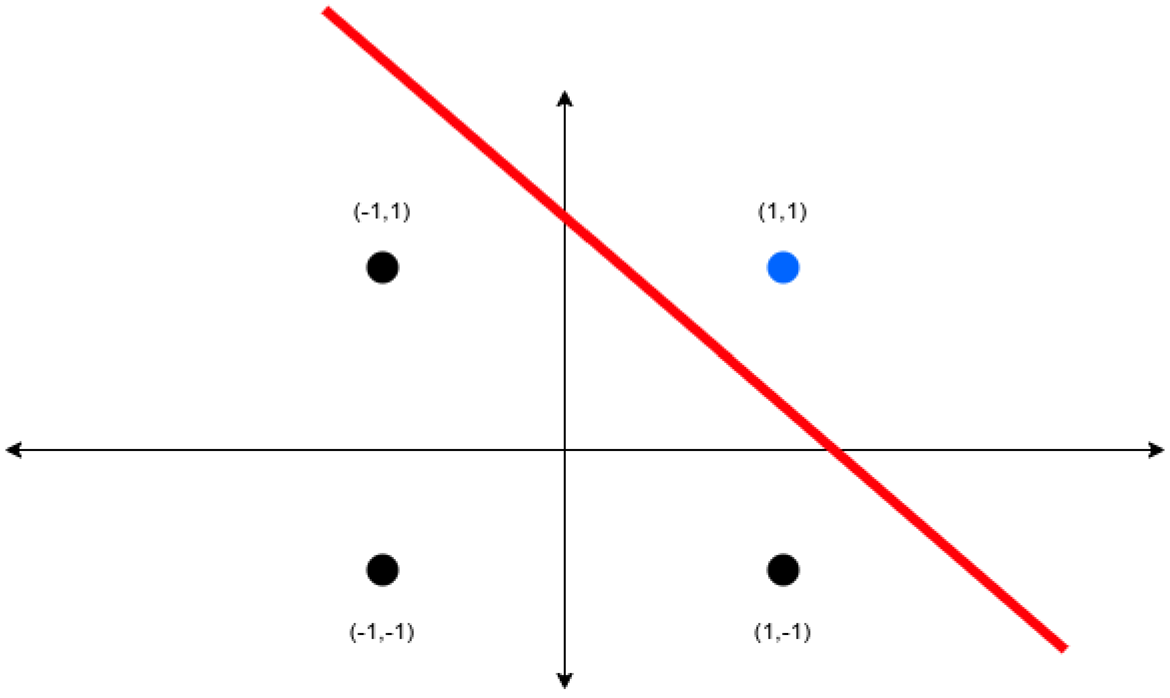
```
import numpy as np

data = np.array([50, 55, 70, 80, 130, 150, 155, 160])
labels = np.array([1, 1, 1, 1, -1, -1, -1, -1])

trial = 0
error = 1
while(error):
    weight = np.random.rand(2) * 2 - 1
    error = np.sum(np.sign(weight[0]*data+weight[1]) - labels)
    trial += 1

print("Number of trials:", trial)
```

Q3) Given a data set of 2D points, {(-1, -1), (+1, -1), (-1, +1), (+1, +1)} and their corresponding labels {+1, +1, +1, -1}, find and visualize a classification boundary assuming the use of a bipolar perceptron with bias.

The input data here is two dimensional so we will be using a two input bipolar perceptron model



And we can conclude from the drawing that the classification line has a perpendicular vector of [-1, -1] that points to the direction of growth (> 0) with a bias of 0.5



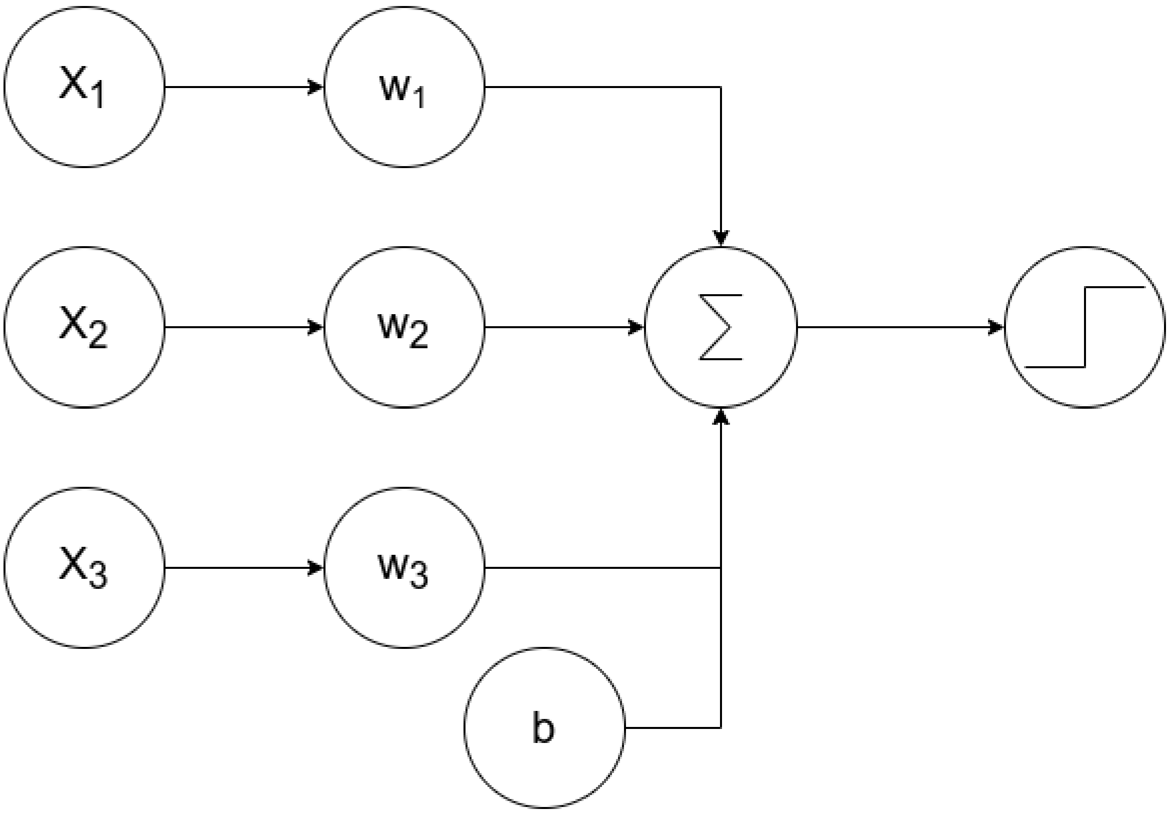
We substitute our assumptions in the classification model with the 4 samples provided:

$$F(x_1, x_2) \ = \ Sign(w_1 * x_1 + w_2 * x_2 + b)$$

1. Sign((-1*-1) + (-1*-1) + 0.5) = 1 …… Correct
2. Sign((-1*1) + (-1*-1) + 0.5) = 1 …… Correct
3. Sign((-1*-1) + (-1*1) + 0.5) = 1 …… Correct
4. Sign((-1*1) + (-1*1) + 0.5) = -1 …… Correct

Q4) Given a data set of RGB colors, {(0, 0, 0), (255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (0, 255, 255), (255, 0, 255), (255, 255, 255)} and their corresponding labels {+1, +1, +1, -1, +1, -1, -1, +1}, find and visualize a classification boundary assuming the use of a bipolar perceptron with bias.

Here, we use a three input bipolar perceptron model like the one shown below

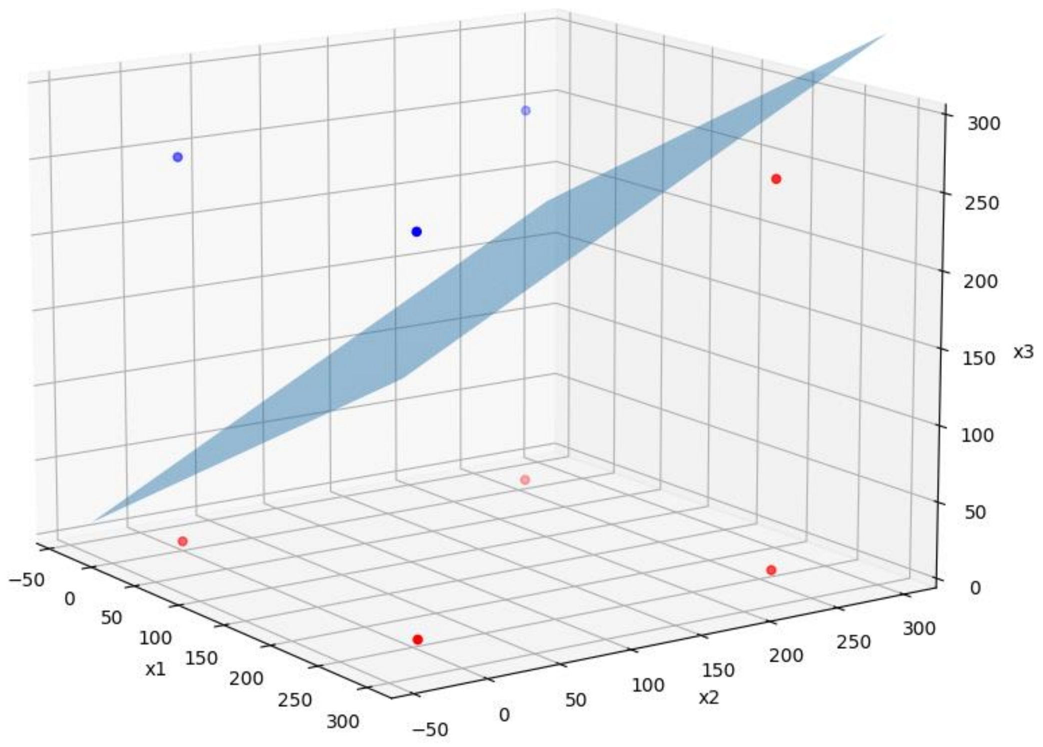


And with the help of visualizing the data points in 3D, we can make a good assumption for the weights and bias … w1 = 1, w2 = 1, w3 = -2 & bias = 100

$$F(x_1, x_2) \ = \ Sign(w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + b)$$

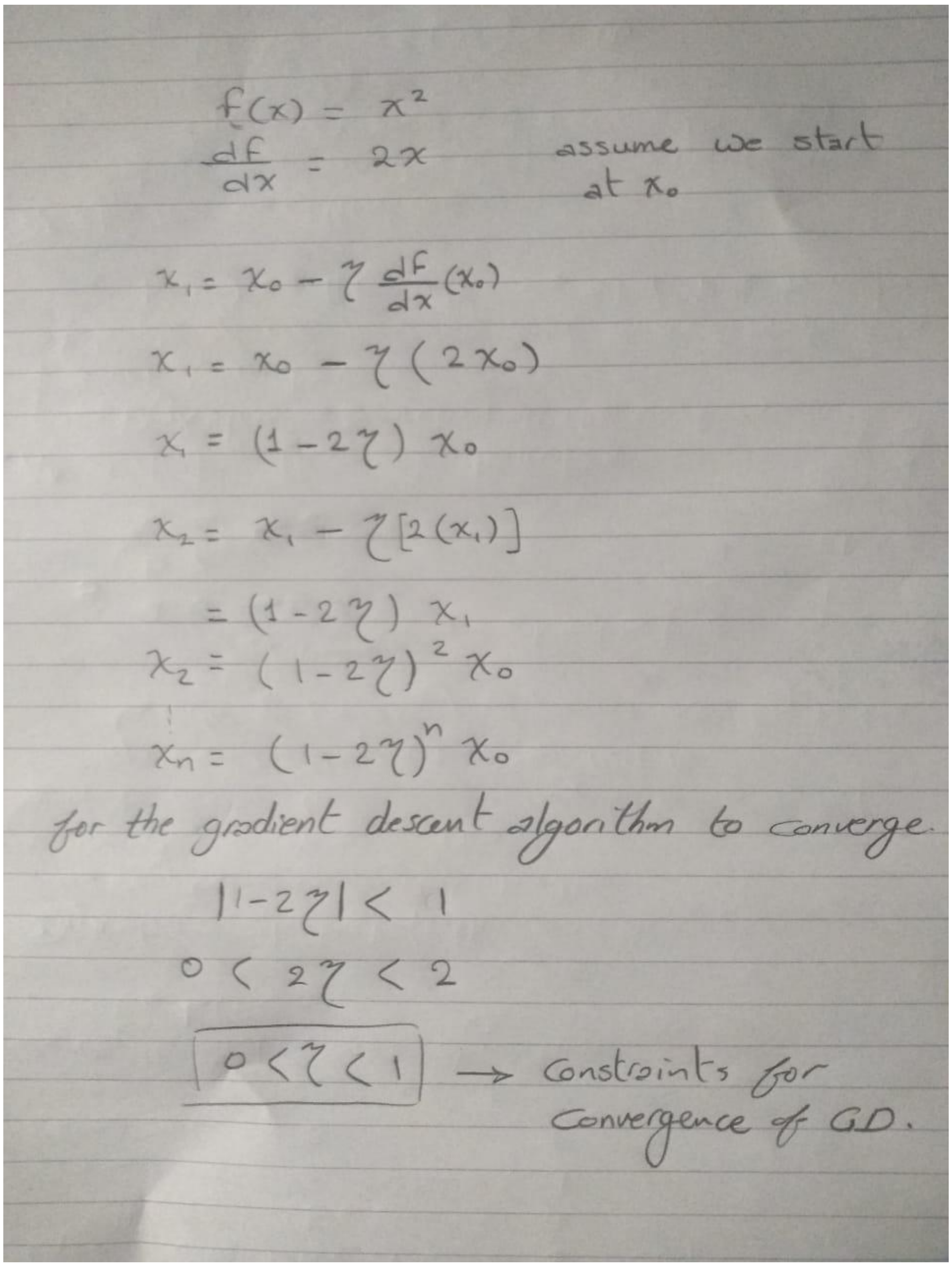
1. Sign((1*0) + (1*0) + (-2*0) + 100) = 1 …… Correct
2. Sign((1*255) + (1*0) + (-2*0) + 100) = 1 …… Correct
3. Sign((1*0) + (1*255) + (-2*0) + 100) = 1 …… Correct
4. Sign((1*0) + (1*0) + (-2*255) + 100) = -1 …… Correct

5. Sign((1*255) + (1*255) + (-2*0) + 100) = 1 …… Correct
6. Sign((1*0) + (1*255) + (-2*255) + 100) = -1 …… Correct
7. Sign((1*255) + (1*0) + (-2*255) + 100) = -1 …… Correct
8. Sign((1*255) + (1*255) + (-2*255) + 100) = 1 …… Correct



Q5) What is the constraint on the learning rate for minimizing $f(x) = x^2$ using the gradient descent optimization? Use visualization to verify your answer by showing different results for different learning rates w.r.t the optimal learning rate.

Proving learning rate convergence between 0 - 1 with the update rule:



$f(x) = x^2$

$\frac{df}{dx} = 2x$

assume we start at x_0

$x_1 = x_0 - \eta \frac{df}{dx}(x_0)$

$x_1 = x_0 - \eta (2x_0)$

$x_1 = (1 - 2\eta) x_0$

$x_2 = x_1 - \eta [2(x_1)]$

$= (1 - 2\eta) x_1$

$x_2 = (1 - 2\eta)^2 x_0$

$\vdots$

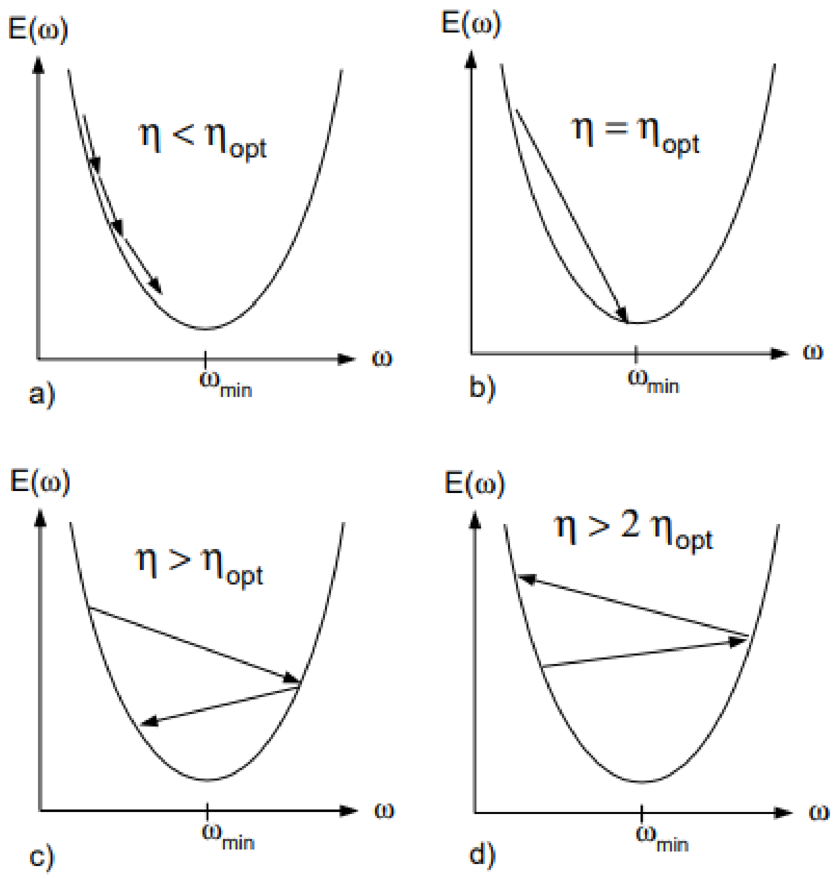
$x_n = (1 - 2\eta)^n x_0$

for the gradient descent algorithm to converge.

$|1 - 2\eta| < 1$

$0 < 2\eta < 2$

$\boxed{0 < \eta < 1} \rightarrow$ Constraints for Convergence of GD.



Q6) Given the following objective function:-

$$F(x,y) = 3x^4 + 3x^2y^2 + x^2 + 2y^4$$

- Use the gradient descent to find a local minimum.
- Use the steepest descent to find a local minimum.
- Use the NR gradient descent to find a local minimum.
- For all of the above cases, use the same initial position and compare your results.
- Verify your solution by visualizing the function surface using your favorite programming language.

1- Vanilla Gradient Descent:

Let initial values be $x_0 = [1,1]$ & $\alpha = 0.05$and $\varepsilon = 0.1$

And First Order Derivatives be

- $F_x = 12x^3 + 6xy^2 + 2x$
- $F_y = 6x^2y + 8y^3$

Compute table with the update rule $X_{n+1} = X_n - \alpha * \nabla f_{x_n}$

X_n	Y_n	$F(X_n, Y_n)$	$\partial F/\partial x$	$\partial F/\partial y$
1	1	9	20	14
0	0.3	0.016	0.0	0.215
0	0.289	0.014	0.0	0.193
0	0.279	0.012	0.0	0.174
0	0.270	0.010	0.0	0.158
0	0.262	0.009	0.0	0.145
0	0.255	0.008	0.0	0.133
0	0.248	0.007	0.0	0.123
0	0.242	0.0069	0.0	0.114
0	0.237	0.0063	0.0	0.106

We will stop here as the next update step will make $||\nabla f(x, y)|| < \varepsilon$

2- Steepest Gradient Descent:

Let initial values be $x_0 = [1, 1]$and $\varepsilon = 0.01$

And First Order Derivatives be

- $F_x = 12x^3 + 6xy^2 + 2x$
- $F_y = 6x^2y + 8y^3$

Update $\alpha^* = argmin\ f(X_i - \alpha * \nabla f_{x_i})$

Compute table with the update rule $X_{n+1} = X_n - \alpha * \nabla f_{x_n}$

X_n	Y_n	$F(X_n, Y_n)$	α	$\partial F/\partial x$	$\partial F/\partial y$
1	1	9	0.052	20	14
-0.041	0.271	0.012	0.66	-0..10	0.162
0.026	0.163	0.002	0.58	0.056	0.035
-0.006	0.142	0.0008	1.0	-0.014	0.023

0.007	0.119	0.0004	0.746	0.01601	0.013
-0.00428	0.109	0.0003	0.981	-0.0088	0.010
0.00442	0.098	0.0002	0.777	0.00911	0.007

We will stop here as the next update step will make $||\nabla f(x,y)|| < \varepsilon$

3- Newton-Raphson Gradient Descent:

Let initial values be $x_0 = [1,1]$and $\varepsilon = 0.1$

And Derivatives be

- $F_x = 12x^3 + 6xy^2 + 2x$
- $F_y = 6x^2y + 8y^3$
- $F_{xx} = 36x^2 + 6y^2 + 2$
- $F_{xy} = 12xy$
- $F_{yx} = 12xy$
- $F_{yy} = 6x^2 + 24y^2$

Construct the Hessian matrix and Compute $\alpha = H^{-1}$

Compute table with the update rule $X_{n+1} = X_n - \alpha * \nabla f_{x_n}$

X_n	Y_n	$F(X_n, Y_n)$	F_x	F_y	F_{xx}	F_{xy}	F_{yx}	F_{yy}
1	1	9	20	14	44	12	12	30
0.632	0.680	1.8648	6.06	4.152	19.1	5.16	5.16	13.50
0.372	0.472	0.3893	1.86	1.236	8.34	2.112	2.112	6.186
0.183	0.337	0.0742	0.56	0.374	3.89	0.741	0.741	2.928
0.056	0.241	0.0105	0.13	0.117	2.46	0.162	0.162	1.418
0.006	0.164	0.0015	0.01	0.035	2.16	0.013	0.013	0.649
0.0003	0.109	0.0002	0.00	0.010	2.07	0.001	0.001	0.289

We will stop here as the next update step will make $||\nabla f(x,y)|| < \varepsilon$

Q7) Use the gradient descent to solve problems (1), (3), (4).

We will use the $max(0, -y_m W^T x_m)$ as a Loss function and its derivative will be $\partial L/\partial W = -y_m x_m$if $y_m W^T x_m \leq 0$ and 0 otherwise (Note $\alpha = 1$& we are using Batch Gradient Descent).

$$w_{i+1} = w_i + y_m x_m$$

Problem 1:

With the initial values be $w = [0\ 0]$ & $delta = [0\ 0]$ the Gradient Descent Algorithm will need 2670 step to reach the solution, because of the huge number of iterations the solver of this sheet decided not to include the table and otherwise include the code for the operation (All codes are available in *sheet2_codes* folder)

Problem 2:

Let initial values be $w = [0\ 0\ 0]$ & $delta = [0\ 0\ 0]$

$delta$	W
[0 0 0]	[0 0 0]
[-0.5 0.5 0.5]	[0.5 -0.5 -0.5]
[0 0 0]	[0.5 -0.5 -0.5]

Problem 3:

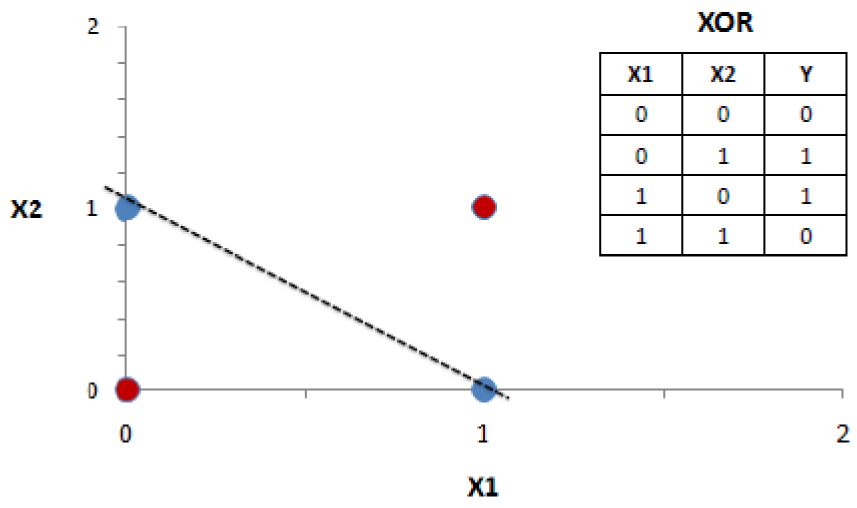
Let initial values be $w = [0\ 0\ 0\ 0]$ & $delta = [0\ 0\ 0\ 0]$

$delta$	W
[0 0 0 0]	[0 0 0 0]
[-0.25 -63.75 -63.75 63.75]	[0.25 63.75 63.75 -63.75]
[0.25 31.875 31.875 63.75]	[0 31.875 31.875 -127.5]
[-0.25 -31.875 -31.875 -31.875]	[0.25 63.75 63.75 -95.625]

[0. 0. 0. 0.]	[0.25 63.75 63.75 -95.625]

Q8) Show that the bipolar perceptron will fail in solving the XOR problem.

With the XOR truth table we find out that it's impossible to separate between its inputs with a one bipolar perceptron model, we will either need a nonlinear model or multiple linear models working together … and here the idea of more than one perceptron one layer Neural Network truly shines.



XOR

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Some History on the XOR problem

In 1969, Minsky and Papert wrote about the shortcomings of the linear models that were adopted heavily in their time **with the XOR problem.** This caused a backlash in the community and was documented in their book Perceptrons: An Introduction to Computational Geometry. And the damage wasn't undone until 1986 when Rumelhart introduced the idea of multilayer perceptron and backpropagation (with it, the first AI winter was seeing some sun).