PERCEPTRON SUB GRADIENT DESCENT

STANDARD GRADIENT DESCENT

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data =pd.read_csv("perceptron.data", names=["X1","X2","X3","X4","Y"])

X = data.loc[:, "X1":"X4"]
Y = data.loc[:, "Y":]
X['Xb'] = 1

step_size = 1
loss = 100
misclassifications = []
missclassification_rate = 1000
W = np.array([0,0,0,0,0]).reshape(1,5)
G = np.array([1,1,1,1,1]).reshape(1,5)
W_list = []
G_list = []
loss_list = []
missclassification_list = []

X = X.values
Y = Y.values

def StandardGradientDescent(step_size, precision):
    global G, W
    iterations = 0
    while abs(G).any() > precision :
        Con = (np.dot(X,W.transpose()))*(-1*Y)
        Con[Con>=0] = 1
        Con[Con<0] = 0
        G = np.sum((Con*Y)*X, axis=0)
        W = W + step_size*(G)
        W_list.append(W.tolist())
        iterations = iterations+1
    return iterations

print(StandardGradientDescent(1,0.01))
```

RESULTS :

```
('ITERATIONS ', 47)
('WEIGHTS AND BIASES (THE LAST TERM IN EACH LIST) FOR FIRST THREE ITERATIONS ', [[[1278.9964610830002, 460.061258012, -108.55851403600002, -1672.3157294789999, -354.0]], [[1307.2947297410003, 432.74778799, -27.55191988300001, -1523.7889544579998, -493.0]], [[1255.1898136200002, 425.504028824, 18.796540398999994, -1434.6675419699998, -625.0]]])
```

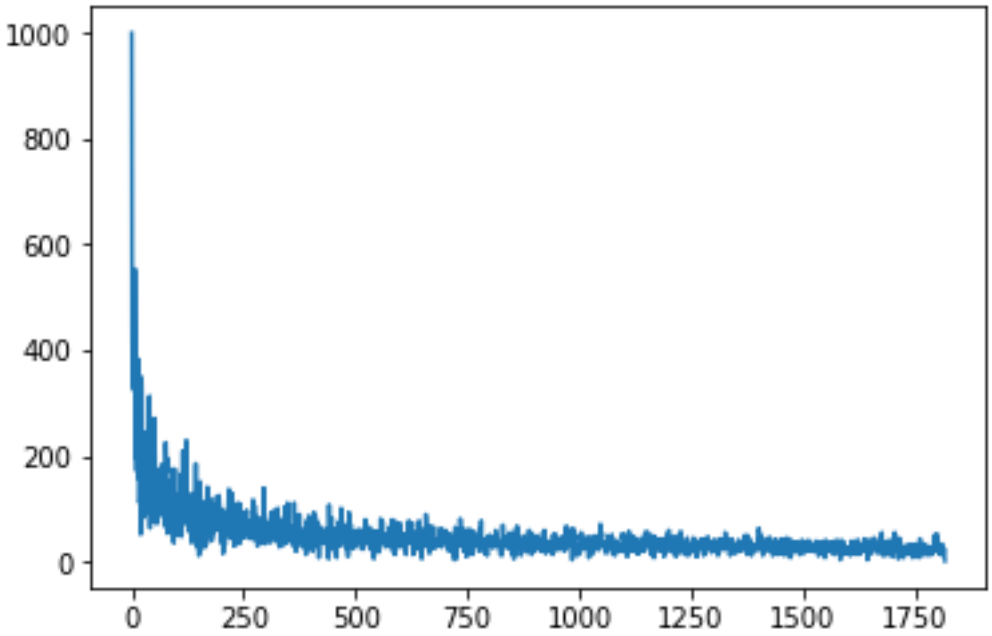
('FINAL WEIGHT AND BIAS : ', array([[685.79932891,   243.89947473,   8.24199193, -797.62505313,
       -1485.   ]]))

STOCHASTIC GRADIENT DESCENT

```
iterations = 0
index = 0
def StochasticGradientDescent(step_size, precision):
    global G, W, loss, missclassification_rate, index, iterations
    pred = []
    while missclassification_rate > 0:
        pred = np.dot(X, W.transpose())
        pred[pred>0] = 1
        pred[pred<0] = -1
        misclassifications = abs(pred - Y)
        missclassification_rate = np.sum(misclassifications)
        missclassification_list.append(missclassification_rate)

        if misclassifications[index%1000] == 0:
            while misclassifications[index%1000] == 0:
                index = index+1

        G = (Y[index%1000]*X[index%1000])

        iterations = iterations+1
        index = index+1

        W = W + step_size*(G)
        W_list.append(W.tolist())

    return iterations

#print(StochasticGradientDescent(1, 0))
#plt.plot(missclassification_list)
```

RESULTS :

('ITERATIONS ', 1818)
('WEIGHTS AND BIASES (THE LAST TERM IN EACH LIST) FOR FIRST THREE ITERATIONS ', [[[4.617544237, 2.4696793809999997, 1.9676607869999998, -1.813355506, -1.0]], [[3.4532228789999997, 0.1694348179999996, 2.62801595, -4.647098506, -2.0]], [[0.45610583999999976, 4.929004722, -2.2588903150000004, -4.651540471, -3.0]]])
('FINAL WEIGHT AND BIAS : ', array([[66.01763148,   24.1532232 ,   -0.75219873, -76.42419023, -142.   ]]))


1000
800
600
400
200
0
0
250
500
750
1000
1250
1500
1750