

Machine Learning (ENME808E): Homework 1

Harsh Kakashaniya(116311236)

February 20, 2019

1 Problem 1

ANSWER :[D]

EXPLANATION:

In first case, we have exact data of variables weight , Size, Denomination so no need of learning as we have exact data. Compare with the given data and classify. So no learning.

In second case, We are defining coins in graph with its features of weight , Size, Denomination and use learning algorithm (E.g. Perceptron) and we can do supervised learning.

In third case, This is an example which learns with the use. Which is an example of Reinforcement learning.

2 Problem 2

ANSWER :[A]

EXPLANATION:

In first case, Classifying numbers into primes and non-primes is a problem with certainty.

In second case, It is a machine learning problem because there will be features which we need to consider for the answer. Hence this can be learned.

In third case, We can certainly define the time. No need of machine learning. Basic physics is involved.

In fourth case, This is also machine learning problem to give optimum results with different parameters consideration which giving output.

3 Problem 3

ANSWER :[D]

EXPLANATION: This is problem of conditional probability.

Let,

B1= Probability of Choosing first black ball.

B2= Probability of Choosing second black ball.

We are computing the following probability.

$$P[B2/B1] = \left(\frac{P[B1 \cap B2]}{P[B1]} \right) \quad (1)$$

$$P[B2/B1] = \frac{(1/2)}{(3/4)} \quad (2)$$

$$P[B2/B1] = 2/3 \quad (3)$$

Which is the answer.

4 Problem 4

ANSWER :[B]

EXPLANATION:

P[Red ball]= 0.55

P[Green ball]=1-P[Red ball]

P[Green ball]=0.45

Probability of no red = Probability of all green

So to get one green we have probability = 0.45

For 10 such marbles.

Because we need all green so we will use 'and' condition to compute the probability.

P[All green]=(0.45)¹⁰

P[All green]=0.0003405

Hence this answer is near to B option

5 Problem 5

ANSWER :[C]

EXPLANATION: Now there are 1000 samples.

$P[\text{At least one sample with no all red}] = P[\text{At least one sample with all green}]$

$P[\text{At least one sample with all green}] = 1 - P[\text{Sample set with no all green}]$

$P[\text{Sample set with No all green}] = (1 - P[\text{All green}]) = (1 - (0.45)^{10})$

$P[\text{At least one sample with no all red}] = 1 - (1 - (0.45)^{10})^{1000}$

$P[\text{At least one sample with no all red}] = 0.2886$

Hence, this answer is near to C option

6 Problem 6

ANSWER :[E]

EXPLANATION: As we are not given with correct target function so there are different target functions which satisfied different hypothesis.

For example

Case 1:

If we have function as

$$X_1 + (\bar{X}_2 + \bar{X}_3) \quad (4)$$

In this case, we will get answer as per hypothesis in answer A which says all remaining 3 y's will yield value 1

Case 2: If we take function as.

$$(\bar{X}_2 + \bar{X}_3) \quad (5)$$

We get output as required in D,

g returns the opposite of the XOR function: if the number of 1s is odd, it returns 0, otherwise returns 1.

Case 3:

if we have hypothesis as:

$if(\bar{X}_2 + \bar{X}_3 == 0) :$

return X_1

else:

return 1

this case will give us output as C option that is

g is the XOR function applied to x, i.e., if the number of 1s in x is odd, g returns 1; if it is even, g returns 0.

Case 4:

$if(X_1 == 1) :$

return $\bar{X}_2 \bar{X}_3$

else:

return $\bar{X}_2 \bar{X}_3$

So all hypothesis is equi probable to be occurring hence we get answer as E.

7 Problem 7

ANSWER :[B]

EXPLANATION: We need to write a code for Perceptron by generation 10 points in random with defined function. Here we have target function as

Target function:

$$x-y+0.2=0$$

So randomly generated points are classified with this constraint.

Output with algorithm running

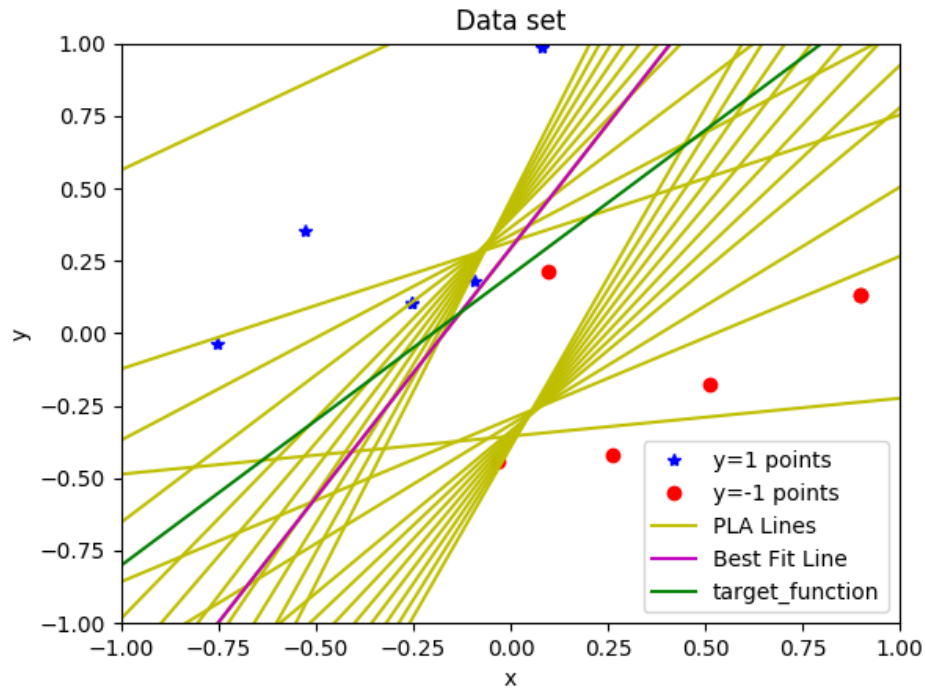


Figure 1: All random lines.

For this we needed 24 lines to classify So nearest answer is 15.

8 Problem 8

ANSWER :[C]

EXPLANATION: We run the code and get probability of miss-classification as follows.

```

hbk@hbk-world: ~/Desktop/Machine Learning/Perceptron_Algorithm
Misclas. pts count 2
Iterations count: 7
Misclas. pts count 1
Iterations count: 8
Misclas. pts count 2
Iterations count: 9
Misclas. pts count 1
Iterations count: 10
Misclas. pts count 1
Iterations count: 11
Misclas. pts count 1
Iterations count: 12
Misclas. pts count 1
Iterations count: 13
Misclas. pts count 1
Iterations count: 14
Misclas. pts count 1
Iterations count: 15
Misclas. pts count 1
Iterations count: 16
Misclas. pts count 1
Iterations count: 17
Misclas. pts count 1
Iterations count: 18
Misclas. pts count 1
Iterations count: 19
Misclas. pts count 2
Iterations count: 20
Misclas. pts count 1
Iterations count: 21
Misclas. pts count 2
Iterations count: 22
Misclas. pts count 1
Iterations count: 23
Misclas. pts count 2
Iterations count: 24
Misclas. pts count 0
x_minus_t
y_minus_h
y_plus_t
y_plus_h
Probability of  $f(x) \neq g(x)$  0.09384120962901471
hbk@hbk-world:~/Desktop/Machine Learning/Perceptron_Algorithm$

```

Figure 2: Output of Probability at N=10

Answer is 0.0938 which is nearly equal to 0.1

9 Problem 9

ANSWER :[B]

EXPLANATION: Same Code if we run for 100 points Changing N=100

Here we have target function as

Target function:

$$x-y+0.2=0$$

So randomly generated points are classified with this constraint.

We get.

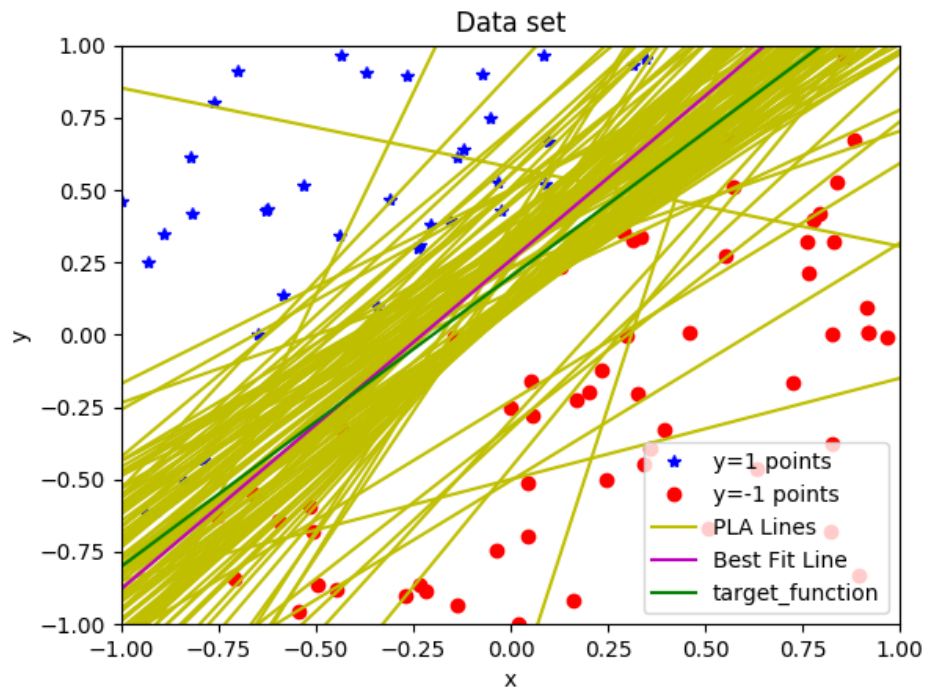


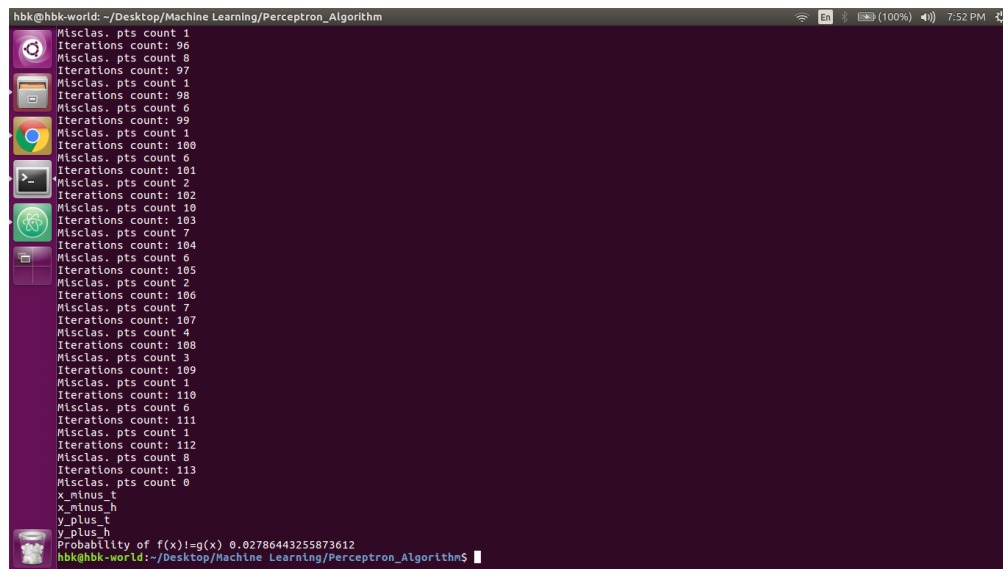
Figure 3: Graph when N=100

And while running code we need 113 iterations which is nearly equal to 100 iterations.

10 Problem 10

ANSWER :[B]

EXPLANATION: So when we take 100 points and calculate the probability of miss-classified points. Following results are obtained.



```
hbk@hbk-world: ~/Desktop/Machine Learning/Perceptron_Algorithm
Misclas. pts count 1
Iterations count: 96
Misclas. pts count 8
Iterations count: 97
Misclas. pts count 1
Iterations count: 98
Misclas. pts count 6
Iterations count: 99
Misclas. pts count 1
Iterations count: 100
Misclas. pts count 6
Iterations count: 101
Misclas. pts count 2
Iterations count: 102
Misclas. pts count 10
Iterations count: 103
Misclas. pts count 7
Iterations count: 104
Misclas. pts count 6
Iterations count: 105
Misclas. pts count 2
Iterations count: 106
Misclas. pts count 7
Iterations count: 107
Misclas. pts count 4
Iterations count: 108
Misclas. pts count 3
Iterations count: 109
Misclas. pts count 1
Iterations count: 110
Misclas. pts count 6
Iterations count: 111
Misclas. pts count 1
Iterations count: 112
Misclas. pts count 8
Iterations count: 113
Misclas. pts count 0
x_minus_t
x_minus_h
y_plus_t
y_plus_h
Probabllity of f(x)!=g(x) 0.02786443255873612
hbk@hbk-world:~/Desktop/Machine Learning/Perceptron_Algorithm$
```

Figure 4: Output of Probability at N=100

when we compute we get probability as 0.027 which is nearly equal to 0.01.

Git hub link for the code:

https://github.com/harshkakashaniya/Perceptron_Algorithm

This is an Github link which has Perceptron with detailed readme and Report

Code is also attached below problem 1.3

11 Problem 1.3

11.1 A part

To Prove: $\rho \geq 0$

Proof:

$$\rho = \min_{(1 \leq n \leq N)} y_n (w^{*T} x_n) \quad (6)$$

we know that all points are classified when we use w^*

So always,

$$\text{sign}(w^{*T} x_n) = y_n \quad (7)$$

Hence we can say,

As, If $w^{*T} x_n$ is positive it implies y_n is positive

and If $w^{*T} x_n$ is negative it implies y_n is negative.

So in both cases,

$y_n (w^{*T} x_n)$ is always positive

$$\rho > 0$$

Hence Proved

11.2 B part

To prove:

$$w^T(t)w^* \geq w^T(t-1)w^* + \rho \quad (8)$$

Proof:

so we have:

$$w^T(t) = w^T(t-1) + x_n^T y_n \quad (9)$$

post multiply both sides w^*

$$w^T(t)w^* = w^T(t-1)w^* + x_n^T y_n w^* \quad (10)$$

so we can write

$$x_n^T y_n w^* = y_n w^{*T} x_n$$

Now we can substitute,

From first condition, For any point

$$y_n w^{*T} x_n \geq \rho$$

Hence we get the following equality

$$w^T(t)w^* \geq w^T(t-1)w^* + \rho$$

secondly,

using induction

$$w^T(t)w^* \geq w^T(t-1)w^* + \rho$$

then, with recursion

$$w^T(t)w^* \geq (w^T(t-2)w^* + \rho) + \rho$$

$$w^T(t)w^* \geq (w^T(t-3)w^* + \rho) + \rho + \rho$$

So this will go on, till we have term $w^T(t-t)w^*$

and we know,

$$w^T(t-t)w^* = 0$$

So in this case we will have t terms of ρ

Hence we get,

$$w^T(t)w^* \geq t\rho \quad (11)$$

Hence Proved

11.3 C part

we have

$$w^T(t) = w^T(t-1) + y(t-1)x(t-1) \quad (12)$$

squaring on both sides

$$w^T(t)^2 = (w^T(t-1) + y(t-1)x(t-1))^2 \quad (13)$$

$$w^T(t)^2 = (w^T(t-1))^2 + (y(t-1)x(t-1))^2 + 2(w^T(t-1)(y(t-1)x(t-1))) \quad (14)$$

Taking norm on both sides.

$$||w^T(t)||^2 \leq ||(w^T(t-1))||^2 + ||(y(t-1)x(t-1))||^2 + 2||w^T(t-1)(y(t-1)x(t-1))||$$

as we have property $||a+b|| \leq ||a|| + ||b||$

Now as we are taking square of norm

$$||y(t-1)x(t-1)||^2 = ||x(t-1)||^2$$

Hence we have

$$||w^T(t)||^2 \leq ||(w^T(t-1))||^2 + ||x(t-1)||^2 + 2||w^T(t-1)(y(t-1)x(t-1))||$$

As t-1 point is miss classified so

As, If $w^{*T}x_n$ is positive it implies y_n is negative

and If $w^{*T}x_n$ is negative it implies y_n is positive.

As sign of both terms will be opposite.

$$w^T(t-1)x(t-1)y(t-1) \leq 0$$

Hence we get the required equality.

$$||w^T(t)^2|| \leq ||(w^T(t-1))||^2 + ||x(t-1)||^2$$

Hence Proved

11.4 D part

From induction.

$$||w^T(t)||^2 \leq ||(w^T(t-2))||^2 + ||x(t-2)||^2 + ||x(t-1)||^2$$

So in such way if we carry on we get

$$||w^T(t)^2|| \leq ||x(0)||^2 + ||x(1)||^2 \dots ||x(t-2)||^2 + ||x(t-1)||^2$$

And we are given $R \geq ||x(0)|| \text{ or } ||x(1)|| \text{ or } \dots \text{ or } ||x(t-1)||$

there are such t terms.

$$||w^T(t)^2|| \leq ||x(0)||^2 + ||x(1)||^2 \dots ||x(t-2)||^2 + ||x(t-1)||^2 \leq tR^2$$

so,

$$||w^T(t)^2|| \leq tR^2$$

Hence Proved

11.5 E part

From b and d

we have

$$w^T(t)w^* \geq t\rho$$

$$||w^T(t)^2|| \leq tR^2$$

Taking root and dividing equality will remain same because Numerator has greater than equality and denominator has less than.

So resultant of the fraction will have greater than and equal to.

$$\frac{w^T(t)w^*}{||w^T(t)||} \geq \frac{t\rho}{\sqrt{t}R}$$

$$\frac{w^T(t)w^*}{||w^T(t)||} \geq \sqrt{t}\frac{\rho}{R}$$

we can transfer R/ρ because they are positive

$$\frac{R}{\rho} \frac{w^T(t)w^*}{||w^T(t)||} \geq \sqrt{t}$$

squaring on both sides.

$$\frac{R^2}{\rho^2} \frac{(w^T(t)w^*)^2}{||w^T(t)||^2} \geq t$$

Now we have property of inequality

$$ab \leq ||a||||b||$$

$$\text{that means } \frac{ab}{||a||||b||} \leq 1$$

$$\frac{R^2}{\rho^2} \frac{||w^T(t)w^*||}{||w^T(t)||} \geq t$$

$$\frac{R^2}{\rho^2} ||w^*||^2 \geq t$$

Hence Proved

12 PLA Code

```
import argparse
import numpy as np
import os, sys
from numpy import linalg as LA
import math
import pickle
import matplotlib.pyplot as plt
import random
N=100
### data generation
#-----
data1=np.column_stack((np.ones(N),np.ones(N),np.ones(N)))
target_fn_m=1
# keep value of c between -1<c<1 because we have points between -1 to 1
target_fn_c=0.2

for i in range(0,len(data1)):

    data1[i,0]=random.uniform(-1,1)
    data1[i,1]=random.uniform(-1,1)
    if (data1[i,0]*target_fn_m+target_fn_c<data1[i,1]):
        a=i
        data1[i,2]=1
    else:
        b=i
        data1[i,2]=-1
#print (data1)
#-----
#data plotting
count=0
break_point=1
for i in range(0,len(data1)):
    if(data1[i,2]==1):
        plt.plot(data1[i,0],data1[i,1], '*b')
    else:
        plt.plot(data1[i,0],data1[i,1], 'or')

plt.plot(data1[a,0],data1[a,1], '*b', label='y=1_points')
plt.plot(data1[b,0],data1[b,1], 'or', label='y=-1_points')
#-----
def ProbabilityError(target_line_m, target_line_c, hypothesis_line_m, hypothesis_line_c):
    x=(hypothesis_line_c-target_line_c)/(target_line_m-hypothesis_line_m)
```

```

y=target_line_m*x+target_line_c
##
y_minus=-1
y_plus=1

x_y_minus_t=(y_minus-target_line_c)/target_line_m
x_y_plus_t=(y_plus-target_line_c)/target_line_m
x_y_minus_h=(y_minus-hypothesis_line_c)/hypothesis_line_m
x_y_plus_h=(y_plus-hypothesis_line_c)/hypothesis_line_m

x_minus=-1
x_plus=1
y_x_minus_t=target_line_m*x_minus+target_line_c
y_x_plus_t=target_line_m*x_plus+target_line_c
y_x_minus_h=hypothesis_line_m*x_minus+hypothesis_line_c
y_x_plus_h=hypothesis_line_m*x_plus+hypothesis_line_c

if(abs(x_y_minus_t)<=1):
    Dty=y_minus
    Dtx=x_y_minus_t
    print( 'y_minus_t ' )

if(abs(y_x_minus_t)<=1):
    Dty=y_x_minus_t
    Dtx=x_minus
    print( 'x_minus_t ' )

if(abs(x_y_minus_h)<=1):
    Dhy=y_minus
    Dhx=x_y_minus_h
    print( 'y_minus_h ' )

if(abs(y_x_minus_h)<=1):
    Dhy=y_x_minus_h
    Dhx=x_minus
    print( 'x_minus_h ' )

#-----
if(abs(x_y_plus_t)<=1):
    Uty=y_plus
    Utx=x_y_plus_t
    print( 'y_plus_t ' )

```

```

if (abs(y_x_plus_t) <= 1):
    Uty=y_x_plus_t
    Utx=x_plus
    print ( 'x_plus_t ' )

```

```

if (abs(x_y_plus_h) <= 1):
    Uhy=y_plus
    Uhx=x_y_plus_h
    print ( 'y_plus_h ' )

```

```

if (abs(y_x_plus_h) <= 1):
    Uhy=y_x_plus_h
    Uhx=x_plus
    print ( 'x_plus_h ' )

```

```

###

```

```

if (abs(x) < 1 and abs(y) < 1):
    intersect = 1
    xm=x
    ym=y
    def Area(x1,y1,x2,y2,x3,y3):
        return abs(0.5*((x1)*(y2-y3)+(x2)*(y3-y1)+(x3)*(y1-y2)))
    Area1=Area(xm,ym,Dtx,Dty,Dhx,Dhy)
    Area2=Area(xm,ym,Utx,Uty,Uhx,Uhy)
    Area=Area1+Area2
else:
    intersect = 0
    def LineLeftArea(Ux,Uy,Dx,Dy):
        if (Ux>Dx):
            Area=abs(abs(Ux+1)*abs(Uy-Dy)-0.5*abs(Ux-Dx)*abs(Uy-Dy))
        else:
            Area=abs(abs(Ux+1)*abs(Uy-Dy)+0.5*abs(Ux-Dx)*abs(Uy-Dy)+abs(Dy+1)*2)
        return Area

    def LineRightArea(Ux,Uy,Dx,Dy):
        if (Ux>Dx):
            Area=abs(abs(Ux-1)*abs(Uy-Dy)+0.5*abs(Ux-Dx)*abs(Uy-Dy)+abs(Dy+1)*2)
        else:
            Area=abs(abs(Ux-1)*abs(Uy-Dy)-0.5*abs(Ux-Dx)*abs(Uy-Dy))
        return Area

```

```

Area1=LineLeftArea(Utx,Uty,Dtx,Dty)
Area2=LineRightArea(Uhx,Uhy,Dhx,Dhy)
#print(Area1,'ar1')
#print(Area2,'ar2')
Area=abs(4-Area1-Area2)

```

```

return Area

```

```

#

```

```

def misclassified(data,w):
    break_point=1
    def safe(num):
        if (num>0):
            return 1
        else:
            return -1
    X=np.column_stack((np.ones(len(data1),dtype=int),data1[:,0],data1[:,1]))
    misclassify=[]
    for i in range(0,len(data1)):
        mul=w*X.transpose()
        sign=safe(float(mul[0,i]))
        if(sign!=data1[i,2]):
            misclassify=np.append([misclassify],[i])
            #print(misclassify,'In loop')
    if(len(misclassify)==0):
        break_point = 0
        a=-5
        length=0
    if(break_point == 1):
        point=random.randint(1,len(misclassify))
        a=int(misclassify[point-1])
        #print("Misclas. pts:",misclassify)
    print("Misclas._pts_count",len(misclassify))
    length=len(misclassify)
    return a,break_point,length

```

```

X=np.column_stack((np.ones(len(data1),dtype=int),data1[:,0],data1[:,1]))

```

```

mis_counter=0

```

```
mis_avg=0
```

```
w=np.mat([0.5,0,0])
```

```
while(break_point==1):
```

```
    count=count+1
```

```
    print("Iterations_count:",count)
```

```
    koyal,break_point,length=misclassified(data1,w)
```

```
    if(koyal!=-5):
```

```
        mis_counter=mis_counter+length
```

```
        #print("Misclas. pts count",mis_counter)
```

```
        w_new=np.mat([0,0,0])
```

```
        w_new= w + X[koyal,:]*data1[koyal,2]
```

```
        #print(w_new)
```

```
        w=w_new
```

```
        m=float(-w_new[0,1]/w_new[0,2])
```

```
        c=float(-w_new[0,0]/w_new[0,2])
```

```
        x_line1=np.mat([[ -1],[1]])
```

```
        y_line1=m*x_line1+c
```

```
        plt.plot(x_line1,y_line1,'-y')
```

```
        plt.xlabel('x')
```

```
        plt.ylabel('y')
```

```
        plt.axis([-1,1,-1,1])
```

```
        plt.title('Data_set')
```

```
        plt.pause(0.01)
```

```
#mis_avg=mis_counter/(len(data1)*count)
```

```
m=float(-w_new[0,1]/w_new[0,2])
```

```
c=float(-w_new[0,0]/w_new[0,2])
```

```
x_line1=np.mat([[ -1],[1]])
```

```
y_line1=m*x_line1+c
```

```
x_linet=x_line1
```

```
y_linet=target_fn_m*x_linet+target_fn_c
```

```
Area=ProbabilityError(target_fn_m,target_fn_c,m,c)
```

```
# as total area of plot is 2x2 which is 4 so we substract area by 4 to get probability
```

```
print('Probability_of_f(x)!=g(x)',Area/4)
```

```
plt.plot(x_line1,y_line1,'-y',label='PLA_Lines')
```

```
plt.plot(x_line1,y_line1,'-m',label='Best_Fit_Line')
```

```
plt.plot(x_linet,y_linet,'-g',label='target_function')
```



```
plt.xlabel('x')
plt.ylabel('y')
plt.axis([-1,1,-1,1])
plt.title('Data_set')
plt.legend()
plt.show()
```