Machine Learning ENME808E - Homework 1

Koyal Bhartia - 116350990

February 20, 2019

1 Question 1

What types of Machine Learning, if any, best describe the following three scenarios:
(i) A coin classification system is created for a vending machine. The developers obtain exact coin specifications from the U.S. Mint and derive a statistical model of the size, weight, and denomination, which the vending machine then uses to classify coins.
(ii) Instead of calling the U.S. Mint to obtain coin information, an algorithm is presented with a large set of labeled coins. The algorithm uses this data to infer decision boundaries which the vending machine then uses to classify its coins.
(iii) A computer develops a strategy for playing Tic-Tac-Toe by playing repeatedly and adjusting its strategy by penalizing moves that eventually lead to losing.

Options:
[a] (i) Supervised Learning, (ii) Unsupervised Learning, (iii) Reinforcement Learning
[b] (i) Supervised Learning, (ii) Not learning, (iii) Unsupervised Learning
[c] (i) Not learning, (ii) Reinforcement Learning, (iii) Supervised Learning
[d] (i) Not learning, (ii) Supervised Learning, (iii) Reinforcement Learning
[e] (i) Supervised Learning, (ii) Reinforcement Learning, (iii) Unsupervised Learning

Answer:
Analyzing each of the scenarios above as follows:
(i) Here the exact specifications of the coin is provided to the developers. There is no new data for which the model needs learning to do estimation. Hence there is no learning happening.
(ii) The presentation of the large set of "labeled" coins calls for Supervised learning. The algorithm uses this labelling as the supervision to train itself to make further estimations.
(iii) The computer plays repeatedly and learns in every iteration. This learning is based on repeated and past events which determines it's future actions. This is known as reinforcement learning.

Thus the answer to the above question is:
Option (d) : (i) Not learning, (ii) Supervised Learning, (iii) Reinforcement Learning

2 Question 2

Which of the following problems are best suited for Machine Learning?
(i) Classifying numbers into primes and non-primes.
(ii) Detecting potential fraud in credit card charges.
(iii) Determining the time it would take a falling object to hit the ground.
(iv) Determining the optimal cycle for traffic lights in a busy intersection.

Options:
[a] (ii) and (iv)
[b] (i) and (ii)
[c] (i), (ii), and (iii)
[d] (iii)
[e] (i) and (iii)

Answer:
Analyzing each of the problems listed above:
(i) There is no uncertainty or learning involved in classifying nos into primes and non-primes. Prime or non prime can easily be figured out with a division by 2 and there is no training of a machine required.
(ii) As the number of payment methods is increasing, criminals find it easier to detect loopholes in the process and fraud in the payment of credit card charges. There is a lot of uncertainty involved, for example, what category of customers tend to be more involved in fraud etc etc, which needs a lot of learning and estimation.
(iii) The amount of time a falling object takes to hit the ground depends on factors such as the properties of the object, the medium through which it is falling etc. It is a physics problem, and hence no machine learning required. A well defined formula exists to calculate the required time which can easily be implemented.
(iv) The no of vehicles on a road, and hence traffic is not always the same. It depends on the time of the day, the area etc. For example the traffic is high during office/school coming and going hours, and less during the afternoon and at night. Similarly main road will always tend to have more traffic than the inner residential areas. Hence the cycle of traffic lights is a learning process based on the change in traffic during the course of the day. Thus the answer to the above question is:
Option (a) : (ii) and (iv)

3 Question 3

We have 2 opaque bags, each containing 2 balls. One bag has 2 black balls and the other has a black ball and a white ball. You pick a bag at random and then pick one of the balls in that bag at random. When you look at the ball, it is black. You now pick the second ball from that same bag. What is the probability that this ball is also black? **Options**
[a] 1/4
[b] 1/3
[c] 1/2
[d] 2/3
[e] 3/4

Answer:
The above is the problem of conditional probability. Dividing the same into sub events as follows:

Let A - Event of picking the first ball as black and
B - Event of picking even the seconf ball as black
Probability of picking the 1st bag: 1/2
Probability that the black ball is picked after picking the first bag: 1
Probability of picking the 2nd bag: 1/2
Probability that the black ball is picked after picking the second bag: 1/2 Thus, using the basic rules of AND and OR in Probability, we get: P(A) = (1/2)(1)+(1/2)(1/2)=3/4 Probability of the 2nd ball being black is the same as the probability of picking the 1st bag, as the 2nd ball in the 2nd bag is not black. Thus, P(B)=P(Picking 1st bag) = 1/2 Thus the probability of both event A and event B occurring is P[B/A] Using the formula for conditional probability:

$$P[B/A] = \frac{P[A \cap B]}{P[A]} \tag{1}$$

$$P[B/A] = \frac{1/2}{3/4} \tag{2}$$

$$P[B/A] = 2/3 \tag{3}$$

Thus the answer is:
Option (d) : 2/3

4 Question 4

Consider a sample of 10 marbles drawn from a bin containing red and green marbles. The probability that any marble we draw is red is $\mu = 0.55$ (independently, with replacement). We address the probability of getting no red marbles ($\gamma = 0$) in the following cases: We draw only one such sample. Compute the probability that ($\gamma = 0$). The closest answer is ('closest answer' means: |your answer given option| is closest to 0): **Options:**
[a] 7.331 10 6
[b] 3.405 10 4
[c] 0.289
[d] 0.450
[e] 0.550 **Answer:** Let R indicate drawing red balls; and G indicate the drawing of green balls. It is given that probability of drawing a red ball ie. P(R)=0.55. As there are only red and green balls, thus the drawing of green balls i.e P(G) = 1- P(R) = 1-0.55 = 0.45. Probability that all the balls in the sample is not red means that all the balls are green. i.e $P(G)^{10} = (0.45)^{10} = 0.0003405 = 3.405 * 10^{-4}$.

Thus the answer is **Option [b]**.

5 Question 5

We draw 1,000 independent samples. Compute the probability that (at least) one of the samples has ($\gamma = 0$). The closest answer is:
[a] $7.331 * 10^6$

[b] $3.405 * 10^4$
[c] 0.289
[d] 0.450
[e] 0.550
Answer:
The above question uses the concept that P(at least A happening) = 1-P(A not happening). The probability that at least 1 sample has ($\gamma = 0$) means prob of at least 1 sample having no red or all green, which according to the above definition means at P[At least one sample with no all red] = P[At least one sample with all green] Using the laws of basic probability: P[At least one sample with all green]=1 - P[Sample set with no all green]

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

P[At least one sample with not all balls red]=1-$(1 - (0.45)^{10})^{1000} = 0.2886$ This is closest to 0.289 among the given options.

Thus the answer is **Option [c]**.

6 Question 6

Which hypothesis g agrees the most with the possible target functions in terms of the above score? Several target functions are given for the remaining 3; hence the different possible hypothesis is as follows:

Case 1: $X_1 + (\bar{X_2} + \bar{X_3})$

Here all the remaining 3 hypothesis gives the output as 1.

Case 2: $(\bar{X_2} + \bar{X_3})$ Here we get the opposite of the XOR function.

Case 3: $if(\bar{X_2} + \bar{X_3} == 0)$ Hypothesis: X_1 else 1. This is the same as the XOR function.

Case 4: $if(X_1 == 1)$: Hypothesis: $\bar{X_2}\bar{X_3}$ else $\bar{X_2}\bar{X_3}$

Hence all hypothesis are equiprobable to be occurring. Thus the answer is **Option [e]**.

7 Problem 7

Perception Learning Algorithm

Take N = 10. How many iterations does it take on average for the PLA to converge for N = 10 training points?

Answer: The code for the PLA Algorithm has been written from scratch in Python. Generation of the points has been done using the Random function. The target function has been chosen as follows: x-y+0.2=0 Classification of the points has been done based on the target function. Points above this line has been classified as output y=1 and below the line as y=-1. The best fit line has been found by continuously updating the weights and finally finding the optimum set of weights, as shown below.

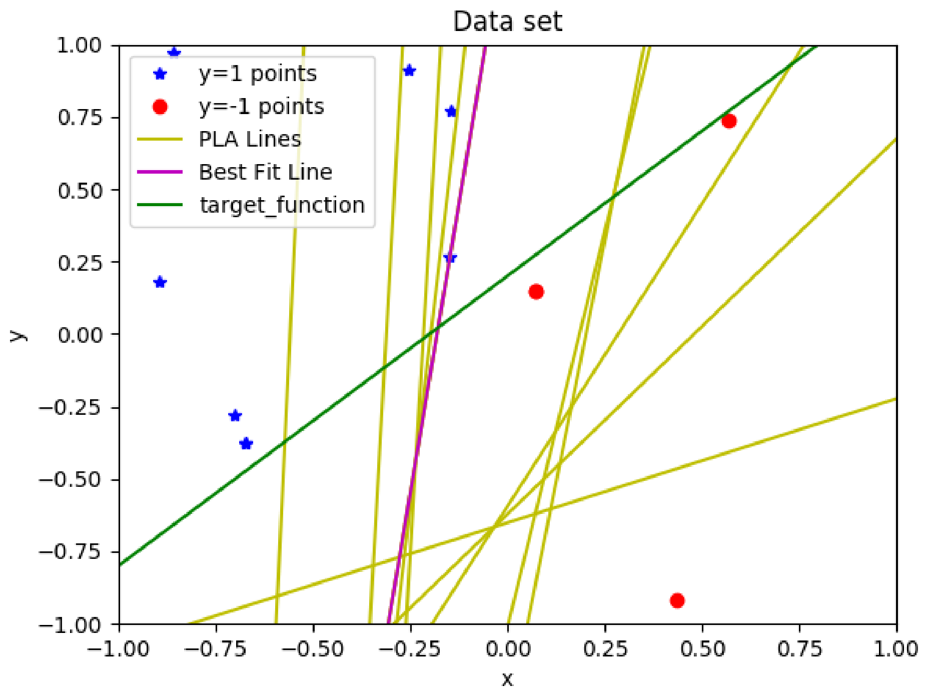
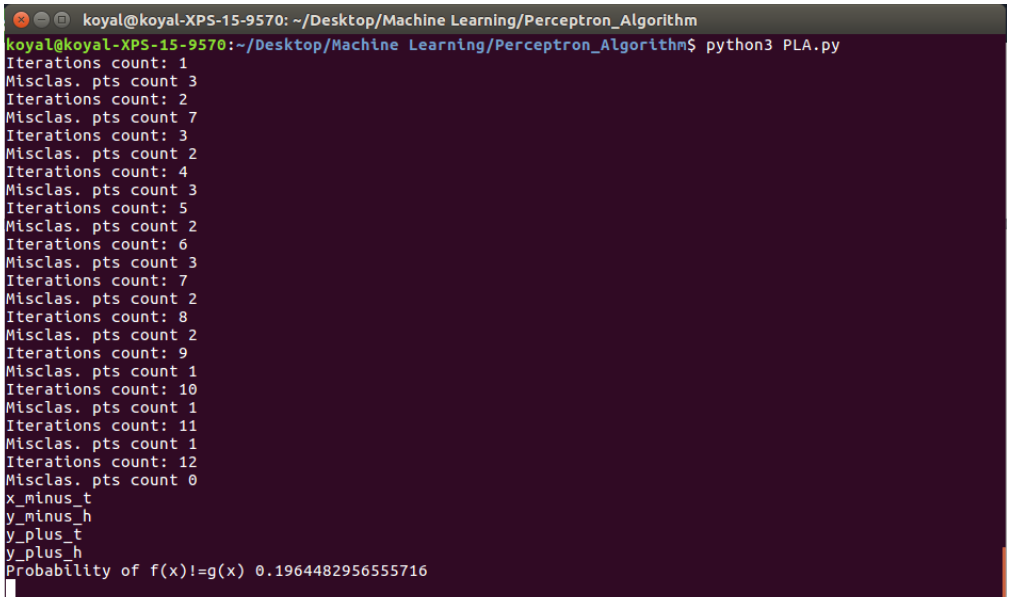


Figure 1: PLA lines and the best fit for 10 Training points.



```
koyal@koyal-XPS-15-9570: ~/Desktop/Machine Learning/Perceptron_Algorithm
koyal@koyal-XPS-15-9570:~/Desktop/Machine Learning/Perceptron_Algorithm$ python3 PLA.py
Iterations count: 1
Misclas. pts count 3
Iterations count: 2
Misclas. pts count 7
Iterations count: 3
Misclas. pts count 2
Iterations count: 4
Misclas. pts count 3
Iterations count: 5
Misclas. pts count 2
Iterations count: 6
Misclas. pts count 3
Iterations count: 7
Misclas. pts count 2
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 0
x_minus_t
y_minus_h
y_plus_t
y_plus_h
Probability of f(x)!=g(x) 0.1964482956555716
```

Figure 2: Terminal output for 10 points

As seen from the terminal screenshot, the no of iterations required for the PLA to converge for 10 training points is 12; which is closest to 15. **Hence the answer is Option[b].**

8 Question 8

Which of the following is closest to P[f (x) ! = g(x)] for N = 10?

Answer:

The difference between f(x) and g(x) has been calculated by taking the diff between the areas occupied by f(x) and g(x). The probability is defined by dividing the difference by the total area i.e 4

As seen in the screenshot provided above, P[f(x)!=g(x)] is found as 0.19 which is closest to 0.1 among the given options. **Hence the answer is Option[b].**

9 Question 9

Now, try N = 100. How many iterations does it take on average for the PLA to converge for N = 100 training points?

Answer:

Now, the same PLA algorithm has been repeated for 100 training points. The data has again been generated randomly, and classified according to the target function:x-y+0.2=0; as already told above.

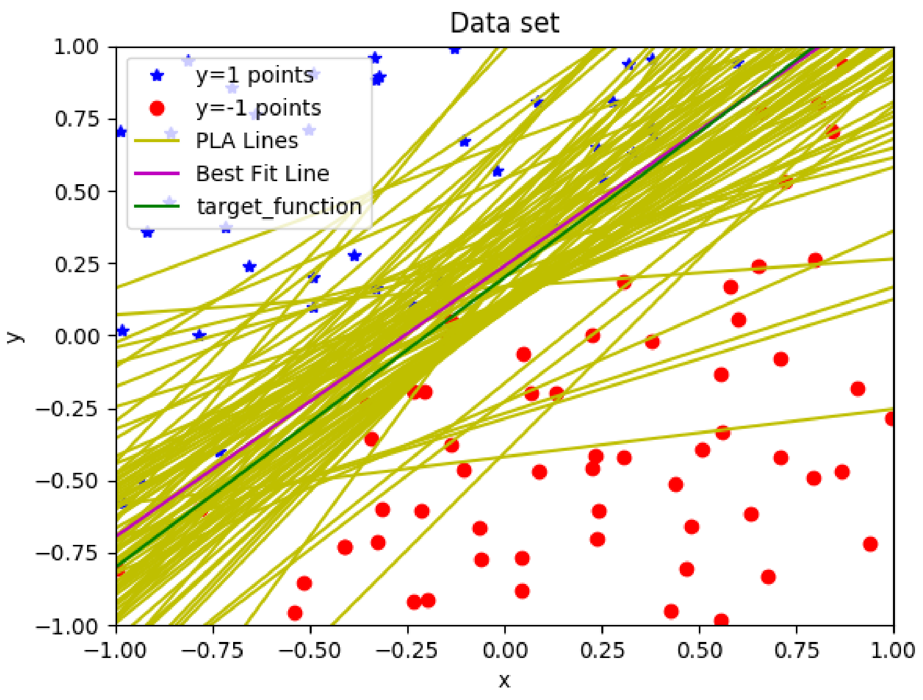
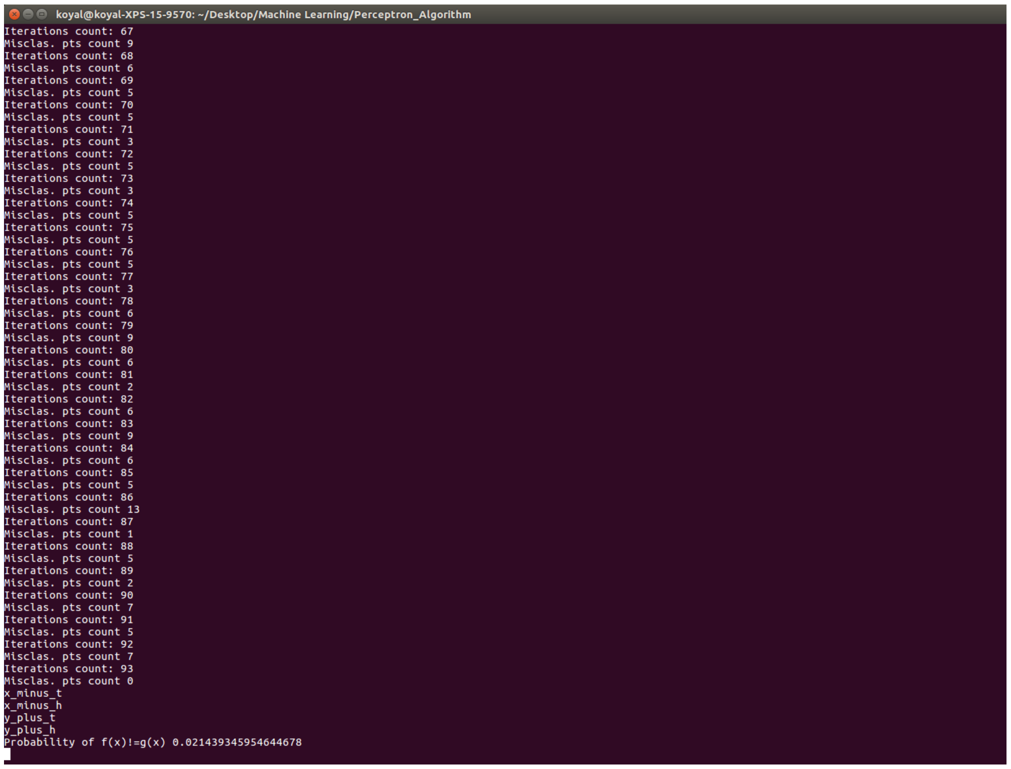


Figure 3: PLA lines and the best fit for 100 Training points.



```
koyal@koyal-XPS-15-9570: ~/Desktop/Machine Learning/Perceptron_Algorithm
Iterations count: 67
Misclas. pts count 9
Iterations count: 68
Misclas. pts count 6
Iterations count: 69
Misclas. pts count 5
Iterations count: 70
Misclas. pts count 5
Iterations count: 71
Misclas. pts count 3
Iterations count: 72
Misclas. pts count 5
Iterations count: 73
Misclas. pts count 3
Iterations count: 74
Misclas. pts count 5
Iterations count: 75
Misclas. pts count 5
Iterations count: 76
Misclas. pts count 5
Iterations count: 77
Misclas. pts count 3
Iterations count: 78
Misclas. pts count 6
Iterations count: 79
Misclas. pts count 9
Iterations count: 80
Misclas. pts count 6
Iterations count: 81
Misclas. pts count 2
Iterations count: 82
Misclas. pts count 6
Iterations count: 83
Misclas. pts count 9
Iterations count: 84
Misclas. pts count 6
Iterations count: 85
Misclas. pts count 5
Iterations count: 86
Misclas. pts count 13
Iterations count: 87
Misclas. pts count 1
Iterations count: 88
Misclas. pts count 5
Iterations count: 89
Misclas. pts count 2
Iterations count: 90
Misclas. pts count 7
Iterations count: 91
Misclas. pts count 5
Iterations count: 92
Misclas. pts count 7
Iterations count: 93
Misclas. pts count 0
x_minus_t
x_minus_h
y_plus_t
y_plus_h
Probability of f(x)!=g(x) 0.021439345954644678
```

Figure 4: Terminal output for 100 points

As seen from the terminal screenshot, the no of iterations required for the PLA to converge for 100 training points is 93; which is closest to 100. **Hence the answer is Option[b].**

10 Question 10

Which of the following is closest to P[f (x)! = g(x)] for N = 100? The difference between f(x) and g(x) has been calculated by finding the difference between the areas and dividing by the total area as explained for the N=10 training points. As seen in the screenshot of the Terminal output, the probability is 0.02 which is closest to 0.01 among the given options. **Hence the answer is Option[b].**
Note: The PLA code is a combined effort of Harsh and I.
Following is the GITHUB link for the code: `https://github.com/Koyal-Bhartia/Perceptron_Algorithm`
The code has also been provided at the end of the document.

11 Problem 1.3

11.1 Let $\rho = min_{(1 \leq n \leq N)} y_n(w^{*T} x_n)$. Show that $\rho > 0$

Proof: The PLA algorithm states that if w^* is the optimal set of weights, then every point is said to be classified and the following thus holds true:

$$sign(w^{*T} x_n) = y_n \tag{4}$$

where x_n is the input data and y_n is the corresponding output. Thus, y_n and $(w^{*T} x_n)$ always hold the same sign, which is similar to square of y_n. As the square of a number is always greater than 0, $y_n(w^{*T} x_n)$ will always be positive and thus $\rho = min_{(1 \leq n \leq N)} y_n(w^{*T} x_n)$ or $\rho > 0$ always. Hence proved.

11.2 Show that $w^T(t)w^* \geq w^T(t-1)w^* + \rho$, and conclude that $w^T(t)w^* \geq t\rho$.

From the definition of PLA algorithm we know that:

$$w^T(t) = w^T(t-1) + x_n^T y_n \tag{5}$$

Post multiplying both sides with w^* we get:

$$w^T(t)w^* = w^T(t-1)w^* + x_n^T y_n w^* \tag{6}$$

$x_n^T y_n w^*$ can also be written as $y_n {w^*}^T x_n$

From the first part of the problem we know that for any point from 1 to N $y_n {w^*}^T x_n \geq \rho$

Substituting both of these in eqn(6) we get the following inequality:

$$w^T(t)w^* \geq w^T(t-1)w^* + \rho \tag{7}$$

Hence proved.

Now, to conclude $w^T(t)w^* \geq t\rho$, we use the method of induction: Using eqn(4) we can write the following:

$$w^T(t)w^* \geq (w^T(t-2)w^* + \rho) + \rho \tag{8}$$

Using recursion we get:

$$w^T(t)w^* \geq (w^T(t-3)w^* + \rho) + \rho + \rho \tag{9}$$

Continuing like this, we get:

$$w^T(t)w^* \geq w^T(t-t)w^* + \rho + \rho ... + \rho(tterms) \tag{10}$$

It is given that w(0)=0, hence the above inequality becomes:

$$w^T(t)w^* \geq t\rho \tag{11}$$

Hence Proved

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

From PLA it is known that:

$$w^T(t) = w^T(t-1) + y(t-1)x(t-1) \tag{12}$$

squaring on both sides gives

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

or

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

Now taking norm on both sides and using the property that $||a+b|| \leq ||a|| + ||b||$ we get:

$$||w^T(t)||^2 \leq ||(w^T(t-1)||^2 + ||(y(t-1)x(t-1))^2|| + 2||(w^T(t-1)(y(t-1)x(t-1)))|| \tag{15}$$

Now as y(t) is +1/-1, $y(t-1)^2$ will always be 1 and hence the following can be implied.

$$||(y(t-1)x(t-1))^2|| = ||x(t-1))^2|| = ||x(t-1)||^2 \tag{16}$$

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

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

Hence the above inequality becomes:

$$||w^T(t)||^2 \leq ||(w^T(t-1)||^2 + ||x(t-1)||^2 + anegativeterm \tag{17}$$

The inequality still holds good; infact more stronger if the negative term is removed.

Hence we get the required equality stating:

$$||w^T(t)||^2 \leq ||(w^T(t-1)||^2 + ||x(t-1)||^2 \tag{18}$$

11.4 Show by induction that $||w^T(t)||^2 \leq tR^2$, where $R = max_{(1 \leq n \leq N)}||x_n||$

From the above proof we have that:

$$||w^T(t)||^2 \leq ||(w^T(t-1)||^2 + ||x(t-1)||^2 \tag{19}$$

Replacing $||(w^T(t-1)||^2$ with the above equation again we get

$$||w^T(t)^2|| \leq ||(w^T(t-2)||^2 + ||x(t-2)||^2 + ||x(t-1)||^2 \tag{20}$$

Similarly replacing each w(t) for t terms, we get

$$||w^T(t)^2|| \leq ||w(0)||^2 + ||x(0)||^2 + ||x(1)||^2 ... ||x(t-2)||^2 + ||x(t-1)||^2 \tag{21}$$

Given that w(0)=0, the above equals:

$$||w^T(t)^2|| \leq ||x(0)||^2 + ||x(1)||^2 ... ||x(t-2)||^2 + ||x(t-1)||^2 \tag{22}$$

Now, it is given that $R = max_{(1 \leq n \leq N)} ||x_n||$ Thus,

$$||w^T(t)^2|| \leq ||x(0)||^2 + ||x(1)||^2 ... ||x(t-2)||^2 + ||x(t-1)||^2 \leq R^2 + R^2 ... + R^2 (t - terms) \tag{23}$$

Hence proved:

$$||w^T(t)||^2 \leq tR^2 \tag{24}$$

11.5 Using (b) and (d), show that $\frac{w^T(t)w^*}{||w^T(t)||} \geq \sqrt{t}\frac{\rho}{R}$ and hence prove that $t \leq \frac{R^2||w^*||^2}{\rho^2}$

From b and d we have the following 2 inequalities:

$$w^T(t)w^* \geq t\rho; ||w^T(t)^2|| \leq tR^2 \tag{25}$$

Taking root of the second inequality: and then dividing the first by the second, the inequality remains the same as below. This is using basic intentional maths as the inequality of the denominator is opposite to that of the numerator. Thus we have:

$$\frac{w^T(t)w^*}{||w^T(t)||} \geq \frac{t\rho}{\sqrt{t}R} \tag{26}$$

or

$$\frac{w^T(t)w^*}{||w^T(t)||} \geq \sqrt{t}\frac{\rho}{R} = \frac{R}{\rho}\frac{w^T(t)w^*}{||w^T(t)||} \geq \sqrt{t} \tag{27}$$

as R/ρ is positive

Now squaring both sides we get:

$$\frac{R^2}{\rho^2}\frac{(w^T(t)w^*)^2}{||w^T(t)||^2} \geq t; \frac{R^2}{\rho^2}\frac{||w^T(t)w^*||||w^T(t)w^*||}{||w^T(t)||||w^T(t)||} \geq t \tag{28}$$

Now using the property of inequality:

a$b \leq ||a||||b||$ or $\frac{ab}{||a||||b||} \leq 1$. Here a and b is $w^T(t)$ and w^* respectively. Hence here:

$$\frac{w^T(t)w^*}{||w^T(t)||||w^*||} \leq 1 \tag{29}$$

Sustituing eqn(29) in (28) the following inequality still holds true.

$$t \leq \frac{R^2||w^*||^2}{\rho^2} \tag{30}$$

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('Area due to T'Area1)
        #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)
#print('Probability of f(x)!=g(x)',mis_avg)


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)
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()
```