Alexander Xu
Professor Alvarez
ML
9/5/19

1)LFD Problem 1.1

Let E be the event such that the first ball picked is black
Let F be the event such that the second ball picked is black

According to Bayes's theorem, we can derive the probability of given that the first ball is black, then the second ball will also be black by solving for **P**(E|F)

Before solving for **P**(E|F), we must first solve for:

$$\mathrm{P}(\mathrm{E} \cap \mathrm{F}) = probability\ of\ 2\ blacks$$
$$= \frac{1}{2}$$

$$Since\ there\ are\ two\ bags, and\ only\ one\ bag\ contains\ 2\ blacks$$

$$\mathrm{P}(\mathrm{F}) = \text{probability of a black}$$
$$= SUM(P(Black\ from\ bag\ 1), P(Black\ from\ bag\ 2)$$
$$= SUM\left(\frac{1}{4}, \frac{1}{2}\right)$$
$$= \frac{3}{4}$$

According to Bayes Theorem:

$$\mathrm{P}(E|F) = \frac{\mathrm{P}(\mathrm{E} \cap \mathrm{F})}{\mathrm{P}(\mathrm{F})} = \frac{\frac{1}{2}}{\frac{3}{4}} = \mathbf{\frac{2}{3}}$$

2)LFD Problem 1.2

a)

1. Show that regions are separated by a line

$If\ h(x) = +1, then\ w^T(x) > 0\ since\ x\ is\ correctly\ classified$
$If\ h(x) = -1, then\ w^T(x) < 0\ since\ x\ is\ not\ correctly\ classified$

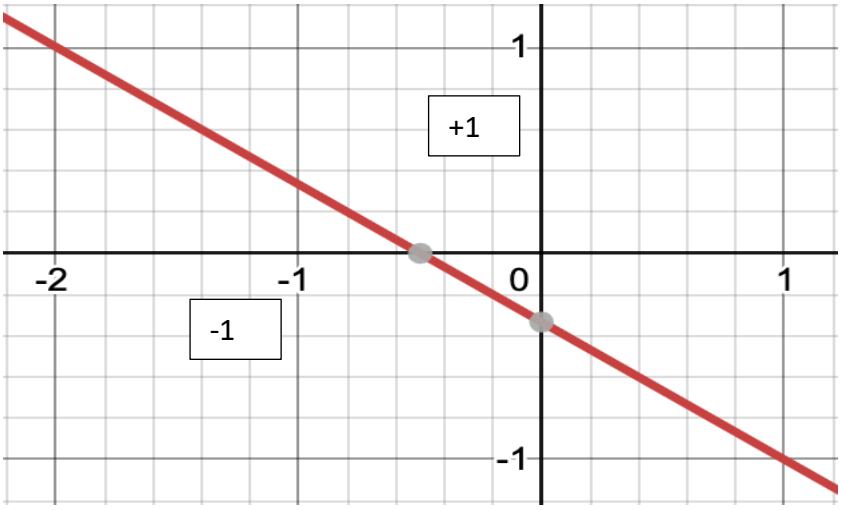
$Since\ h(x) = \pm 1, and\ w^T(x) is\ either > or < than\ 0\ respectively,$
$we\ can\ thereby\ conclude\ that\ the\ two\ regions\ are\ seperated\ by\ a\ line\ such\ that$
$w^T(x) = 0$

2. Calculating slope and intercept in terms of weights

$$w^T(x) = w_2(x_2) + w_1(x_1) + w_0 = 0$$
$$\rightarrow w_2(x_2) = -w_1(x_1) - w_0$$
$$\rightarrow x_2 = \frac{-w_1(x_1)}{w_2} - \frac{w_0}{w_1}$$
$$\boldsymbol{a = \frac{-w_1}{w_2}}$$
$$\boldsymbol{b = -\frac{w_0}{w_1}}$$

b) Graphs

$w = [1,2,3]^T$
$Equation\ of\ Line$:
$y = -\frac{2x}{3} - \frac{1}{3}$
$Graph$:

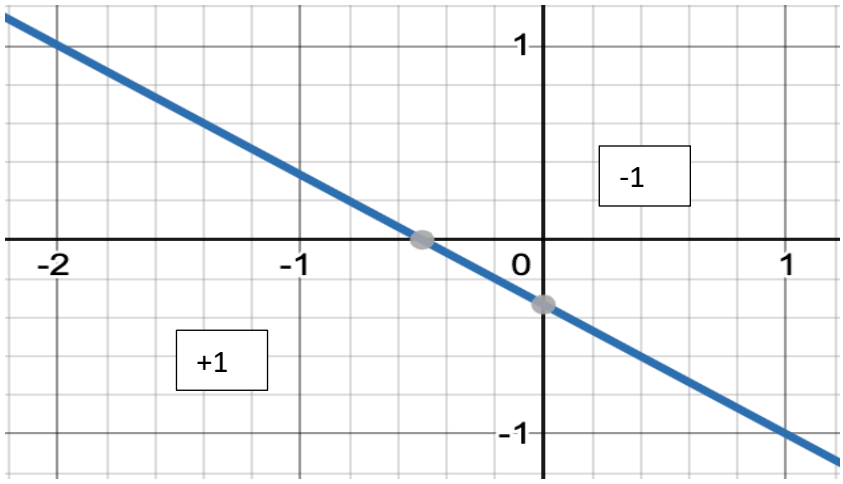


$w = -[1,2,3]^T$

Equation of Line:

$y = -\frac{2x}{3} - \frac{1}{3}$

Graph:



LFD Problem 1.6

a)

Case 1: $\mu = 0.05\ , \gamma = 0.95 \rightarrow 0.95^{10} \approx \mathbf{0.59874}$
Case 2: $\mu = 0.5, \gamma = 0.5 \rightarrow 0.5^{10} \approx \mathbf{9.7656 * 10^{-4}}$
Case 3: $\mu = 0.8, \gamma = 0.2 \rightarrow 0.2^{10} \approx \mathbf{1.024 * 10^{-7}}$

b)

Process:
$= 1 - \mathrm{P}(1000 \text{ samples that has Red})$
$= 1 - [1 - \mathrm{P}(\text{a samples that has } \mathbf{NO} \text{ Red})]^{1000}$
$= 1 - [1 - (1 - \mu)^{10}]^{1000}$

Now we apply the process above to our 3 different cases for μ

Case 1: $\mu = 0.05 \rightarrow 1 - [1 - (1 - 0.05)^{10}]^{1000} \approx \mathbf{1}$
Case 2: $\mu = 0.5 \rightarrow\ 1 - [1 - (1 - 0.5)^{10}]^{1000} \approx \mathbf{0.6235762019}$
Case 3: $\mu = 0.8 \rightarrow\ 1 - [1 - (1 - 0.8)^{10}]^{1000} \approx \mathbf{1.023947625 * 10^{-4}}$

c)

Process:
$= 1 - \mathrm{P}(1000000 \text{ samples that has Red})$
$= 1 - [1 - \mathrm{P}(\text{a samples that has } \mathbf{NO} \text{ Red})]^{1000000}$
$= 1 - [1 - (1 - \mu)^{10}]^{1000000}$

Now we apply the process above to our 3 different cases for μ

Case 1: $\mu = 0.05 \rightarrow 1 - [1 - (1 - 0.05)^{10}]^{1000000} \approx \mathbf{1}$
Case 2: $\mu = 0.5 \rightarrow\ 1 - [1 - (1 - 0.5)^{10}]^{1000000} \approx \mathbf{1}$
Case 3: $\mu = 0.8 \rightarrow\ 1 - [1 - (1 - 0.8)^{10}]^{1000000} \approx \mathbf{0.0973316}$

4) Coding Problems

4a

In [ ]:
```
#Import neccesary packages
from sklearn.linear_model import Perceptron
import numpy as np
import pandas as pd
```

In [ ]:
```
#Loading the data
dataset_1 = np.loadtxt('sampleData1.txt',delimiter=',')
(numSamples_1, numFeatures_1) = dataset_1.shape
data_1 = dataset_1[:,range(2)].reshape((numSamples_1, 2))
labels_1 = dataset_1[:, 2].reshape((numSamples_1,))

dataset_2 = np.loadtxt('sampleData2.txt',delimiter=',')
(numSamples_2, numFeatures_2) = dataset_2.shape
data_2 = dataset_2[:,range(2)].reshape((numSamples_2, 2))
labels_2 = dataset_2[:, 2].reshape((numSamples_2,))

dataset_3 = np.loadtxt('sampleData3.txt',delimiter=',')
(numSamples_3, numFeatures_3) = dataset_3.shape
data_3 = dataset_3[:,range(2)].reshape((numSamples_3, 2))
labels_3 = dataset_3[:, 2].reshape((numSamples_3,))

```

In [ ]:
```
#load perceptron
#generate stop counter
#generate counter to track weight changes
def partial_1():
    perceptron_1 = Perceptron()
    stop_1 = 0
    counter_1 = 0
    while stop_1 == 0:
        for x in range(1000):
            perceptron_1.partial_fit([data_1[x]],[labels_1[x]], classes= np.unique(labels_1))
            if perceptron_1.score(data_1,labels_1) == 1:
                counter_1 +=1
                stop_1 +=1
                break
            else:
                counter_1 +=1
    weights = perceptron_1.coef_
    w1 = weights[0][0]
    w2 = weights[0][1]
    w0 = perceptron_1.intercept_[0]
    print("Weights was adjusted {} times".format(counter_1))
    print("Intercept (w0) is {}".format(w0))
    print("Final weight vector is : {} Hence:".format(weights))
    print("w1 is : {}  , w2 weight is: {}".format(w1, w2))
    #deriving the line based on HW Problem 1.2
    a = -1* (w1/w2)
    b = -1* (w0/w2)
    print("The equation of the decision boundary  line is y = ({})x + ({})".format(a,b))

```

In [ ]:
```
def partial_2():
    perceptron_2 = Perceptron()
    stop_2 = 0
    counter_2 = 0
    while stop_2 == 0:
        for x in range(1000):
            perceptron_2.partial_fit([data_2[x]],[labels_2[x]], classes= np.unique(labels_2))
            if perceptron_2.score(data_2,labels_2) == 1:
                counter_2 +=1
                stop_2 +=1
                break
            else:
                counter_2 +=1
    weights = perceptron_2.coef_
    w1 = weights[0][0]
    w2 = weights[0][1]
    w0 = perceptron_2.intercept_[0]
    print("Weights was adjusted {} times".format(counter_2))
    print("Intercept (w0) is {}".format(w0))
    print("Final weight vector is : {} Hence:".format(weights))
    print("w1 is : {}  , w2 weight is: {}".format(w1, w2))
    #deriving the line based on HW Problem 1.2
    a = -1* (w1/w2)
    b = -1* (w0/w2)
    print("The equation of the decision boundary line is y = ({})x + ({})".format(a,b))

```

In [ ]:

```
def partial_3():
    perceptron_3 = Perceptron()
    stop_3 = 0
    counter_3 = 0
    while stop_3 == 0:
        for x in range(1000):
            perceptron_3.partial_fit([data_3[x]],[labels_3[x]], classes= np.unique(labels_3))
            if perceptron_3.score(data_3,labels_3) == 1:
                stop_3 +=1
            if counter_3 >= 100000:
                return print('No Convergence')
            else:
                counter_3 +=1
    weights = perceptron_3.coef_
    w1 = weights[0][0]
    w2 = weights[0][1]
    w0 = perceptron_3.intercept_[0]
    print("Weights was adjusted {} times".format(counter_3))
    print("Intercept (w0) is {}".format(w0))
    print("Final weight vector is : {} Hence:".format(weights))
    print("w1 is : {}  , w2 weight is: {}".format(w1, w2))
    #deriving the line based on HW Problem 1.2
    a = -1* (w1/w2)
    b = -1* (w0/w2)
    print("The equation of the decision boundary line is y = ({})x + ({})".format(a,b))


```

4b

Main differences:

sampleData3.txt did not converge, which indicates that the points plotted on to a cartesians plane cannot be seperated by a linear line.

As for the main differences between sampleData2.txt and sampleData1.txt, due to the high volume of adjustments in sampleData2.txt, we can assume that the points of true classification and false classifications are tightly plotted. This would account for why the weights had to be adjusted so many times.

w2 has consistently been greater than w1 for the data we tested, which may mean that the second feature played a higher role in determinging the right classification. This seemed to be especially true for sampleData2.txt, with the intercept being 14, we can assume that the line is much higher, which would also explain the high volume of weight adjustments since adjustments are only increased or decreased by a minimal value.

The likely reason why sample_3 doesn't converge is because there doesn't exist a linear line such that positive and negative feedbacks can be distinctly seperated.

In [9]:

```
#Sample 1:
partial_1()
```

```
Weights was adjusted 957 times
Intercept (w0) is 2.0
Final weight vector is : [[ 6.974976 10.593447]] Hence:
w1 is : 6.974976000000002  , w2 weight is: 10.593447
The equation of the decision boundary  line is y = (-0.6584236462409263)x + (-0.18879596037059515)
```

In [10]:

```
#Sample 2:
partial_2()
```

```
Weights was adjusted 56589 times
Intercept (w0) is 14.0
Final weight vector is : [[28.51788  42.613589]] Hence:
w1 is : 28.51788000000003  , w2 weight is: 42.6135890000001
The equation of the decision boundary line is y = (-0.6692203278160862)x + (-0.3285336984875873 4)
```

In [11]:

```
#Sample 3:
partial_3()
```

```
No Convergence
```

5)LFD Problem 1.3

a)

Since w^* is the optimal set
$\rightarrow y_n = \text{sign}(w^* x_n)$
$\rightarrow y_n * (w^* x_n) > 0$ for all n values
We know this to be true since PLA is bounded by the outputs $[-1, 1]$
And since we know the LHS = RHS, any non − zero number multiplied by iself is > 0
$\rightarrow \mathbf{P > 0}$
Since P is merely the minimum of the equation, and above we showed that the equation is greater than 0 for all values of n

b)

$w^T(t)w^*$
$\rightarrow [w^T(t-1) + y(t-1)x(t-1)]w^*$
$\rightarrow w^T(t-1)w^* + y(t-1)x(t-1)w^*$
We know that $y(t-1)x(t-1)w^* \geq P$ since P is the min of the equation
Hence we know that if we add $w^T(t-1)w^*$ to both sides, we get:
$\rightarrow w^T(t-1)w^* + y(t-1)x(t-1)w^* \geq w^T(t-1)w^* + P$
$\rightarrow \mathbf{w^T(t)w^* \geq w^T(t-1)w^* + P}$

Induction:

Base Step: If $t = 0 \rightarrow w^T(0)w^* = 0$
$0P = 0$
$0 \geq 0$

Inductive Step:

Prove: $w^T(t+1)w^* \geq (t+1)P$
$w^T(t+1)w^* \geq w^T(t)w^* + P$
We know that $w^T(t)w^* \geq tP$ in our base step, so if we add + P to both sides:
$w^T(t+1)w^* \geq w^T(t)w^* + P \geq tP + P$
$\rightarrow \mathbf{w^T(t+1)w^* \geq (t+1)P}$

c)

$||w(t)||^2$
$= ||w(t-1)||^2 + 2y(t-1)w^T x(t-1) + ||(y(t-1)x(t-1)||^2$
We know that $2y(t-1)w^T x(t-1) \leq 0$ since $w^T * x(t-1)$ is misclassified. If we remove a number that is ≤ 0 from the RHS:
$||w(t)||^2 \leq ||w(t-1)||^2 + ||y(t-1)x(t-1)||^2$
We know that $y(t-1)$ is restricted to $(-1,1)$, hence $||y(t-1)||^2 = 1$
$\mathbf{||w(t)||^2 \leq ||w(t-1)||^2 + ||x(t-1)||^2}$

d)

Base Step:

$t = 0 \rightarrow \left||w(0)|\right|^2 \leq 0 * R^2 \rightarrow 0 \leq 0$

Induction Step:

Prove: $\left||w(t+1)|\right|^2 \leq (t+1)R^2$

$\left||w(t+1)|\right|^2 \leq \left||w(t)|\right|^2 + ||x(t)||^2$

We know that $\left||w(t)|\right|^2 \leq tR^2$ (from base step), and $||x(t)||^2 \leq R^2$ (since R is the maximum)

$\left||w(t+1)|\right|^2 \leq \left||w(t)|\right|^2 + ||x(t)||^2 \leq tR^2 + R^2$

$\left||w(t+1)|\right|^2 \leq (t+1)R^2$

e)

From b) : $w^T(t)w^* \geq (t)P$

From d) : $\left||w(t)|\right|^2 \leq (t)R^2$

Combining these inequalities, we get:

$\frac{w^T(t)w^*}{\left||w(t)|\right|^2} \geq \frac{(t)P}{(t)R^2}$ (LHS num $>$ RHS num, LHS denom $<$ RHS denom, hence, LHS $\geq$ RHS)

$\rightarrow \frac{w^T(t)w^*}{\left||w(t)|\right|} \geq \frac{(t)P}{\sqrt{t} * R} \rightarrow \frac{w^T(t)w^*}{\left||w(t)|\right|} \geq \sqrt{t} * \frac{P}{R}$

$\rightarrow \frac{(w^T(t)w^*)^2}{\left||w(t)|\right|^2 * \left||w^*|\right|^2} * \left||w^*|\right|^2 \geq t * \frac{P^2}{R^2}$

$\rightarrow \frac{(w^T(t)w^*)^2}{\left||w(t)|\right|^2 * \left||w^*|\right|^2} * \frac{R^2 * \left||w^*|\right|^2}{p^2} \geq t$

$\frac{w^T(t)w^*}{\left||w(t)|\right| * \left||w^*|\right|} \leq 1$ (from the hint) $\rightarrow \frac{(w^T(t)w^*)^2}{\left||w(t)|\right|^2 * \left||w^*|\right|^2} \leq 1$ (Squaring both sides)

$\rightarrow 1 * \frac{R^2 * \left||w^*|\right|^2}{p^2} \geq t$ (substitution)

$\rightarrow \boldsymbol{\frac{R^2 * \left||w^*|\right|^2}{p^2} \geq t}$