P1. D

We can use historical data as training data, ask a ML algorithm to learn from it and make future commercial behavior prediction.

P2. E

$y_{n(t)} \underline{W_{t+1}^T} X_{n(t)} > 0$ (代入 [e] 選項)

$\Rightarrow y_{n(t)}\left(W_t + y_{n(t)} X_{n(t)} \cdot \left\lfloor \frac{-y_{n(t)} W_t^T X_{n(t)}}{\|X_{n(t)}\|^2} + 1 \right\rfloor\right)^T X_{n(t)}$

$\Rightarrow y_{n(t)} W_t^T X_{n(t)} + y_{n(t)}^2 \cdot \|X_{n(t)}\|^2 \cdot \left\lfloor \frac{-y_{n(t)} W_t^T X_{n(t)}}{\|X_{n(t)}\|^2} + 1 \right\rfloor$

令 $0 \leq \varepsilon < 1$，代入 ε 與 $y_{n(t)}^2 = 1$

$\Rightarrow y_{n(t)} W_t^T X_{n(t)} + \|X_{n(t)}\|^2 \cdot \left(\frac{-y_{n(t)} W_t^T X_{n(t)}}{\|X_{n(t)}\|^2} + 1 - \varepsilon \right)$

$\Rightarrow y_{n(t)} W_t^T X_{n(t)} + \left(-y_{n(t)} W_t^T X_{n(t)}\right) + \|X_{n(t)}\|^2 \cdot (1 - \varepsilon)$

$\Rightarrow \|X_{n(t)}\|^2 \cdot (1 - \varepsilon)$

由於 $0 \leq \varepsilon < 1$，所以 $(1-\varepsilon) \geq 0$，且 $\|X_{n(t)}\|^2$ 也大於零

$\Rightarrow \|X_{n(t)}\|^2 \cdot (1 - \varepsilon) > 0$

得證 $\underline{y_{n(t)} W_{t+1}^T X_{n(t)} > 0}$ #

P3. C

設 $\eta(t)$ 為第t次更新時的更新率

inner product

$$w_f^T w_{t+1} \geq w_f^T w_t + \underbrace{\left(\min_n y_n w_f^T x_n\right)}_{\rho} \times \eta(t)$$

length of $\|w_t\|$

$$\|w_{t+1}\|^2 \geq \|w_t\|^2 + \underbrace{\max_n \|x_n\|^2}_{R^2} \times \eta^2(t)$$

Magic Chain!

$$w_f^T w_1 \geq w_f^T w_0 + \rho \times \eta(0)$$

$$w_f^T w \geq w_f^T w + \rho \times \eta(1)$$

$$\vdots$$

$$w_f^T w_T \geq w_f^T w_{T-1} + \rho \times \eta(T-1)$$

$$\Rightarrow w_f^T w_T \geq w_f^T w_0 + \rho \cdot \sum_{i=0}^{T-1} \eta(i)$$

Magic Chain!

$$\|w_1\|^2 \leq \|w_0\|^2 + R^2 \times \eta^2(0)$$

$$\|w_2\|^2 \leq \|w_1\|^2 + R^2 \times \eta^2(1)$$

$$\vdots$$

$$\|w_T\|^2 \leq \|w_{T-1}\|^2 + R^2 \times \eta^2(T-1)$$

$$\Rightarrow \|w_T\|^2 \leq \|w_0\|^2 + R^2 \times \sum_{i=0}^{T-1} \eta^2(i)$$

Start from $\vec{w}_0 = \vec{0}$

$$1 \geq \frac{w_f^T}{\|w_f\|} \cdot \frac{w_T}{\|w_T\|} > \frac{\rho \cdot \sum_{i=0}^{T-1} \eta(i)}{1 \cdot R \sqrt{\sum_{i=0}^{T-1} (\eta(i))^2}} \quad \cdots\cdots \quad ①$$

[b] $\eta(t) = 0.621$

代入式① $\Rightarrow 1 \geq \dfrac{\rho \cdot 0.621 T}{R \cdot \sqrt{0.621^2 \cdot T}} \Rightarrow 1 \geq \dfrac{\rho \sqrt{T}}{R} \Rightarrow \underline{\dfrac{R^2}{\rho^2} \geq T}$ 收斂 #

(第三題續)

[d] $\eta(t)=\frac{1}{1+t}$

$$\sum_{i=0}^{T-1}\eta(i)=1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{T}\cdots\cdots\cdots\textcircled{2}$$

$$\sum_{i=0}^{T-1}\eta^2(i)=1^2+\left(\frac{1}{2}\right)^2+\left(\frac{1}{3}\right)^2+\cdots+\left(\frac{1}{T}\right)^2\cdots\cdots\textcircled{3}$$

代入①式

$$1\geqslant\frac{\rho\cdot\sum_{i=0}^{T-1}\eta(i)}{1\cdot R\sqrt{\sum_{i=0}^{T-1}(\eta(i))^2}}\geqslant\frac{\rho\cdot\left(1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{T}\right)}{R\cdot\left(1^2+\left(\frac{1}{2}\right)^2+\left(\frac{1}{3}\right)^2+\cdots+\left(\frac{1}{T}\right)^2\right)}$$

* 隨著T增加，②、③ 2個數列差距會愈來愈大

當 $T\to\infty$ 時，式② $\to\infty$（發散）而式③ $\simeq 1.6$

由此可知，有某T使 $\frac{\rho\cdot\left(1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{T}\right)}{R\cdot\left(1^2+\left(\frac{1}{2}\right)^2+\cdots+\left(\frac{1}{T}\right)^2\right)}$ 超过1，因此pla会halt.

（第三題續）

[e] $\eta(t) = \left\lfloor \dfrac{-y_{n(t)} W_t^T x_{n(t)}}{\|x_{n(t)}\|^2} + 1 \right\rfloor$

令 $1 \leq \varepsilon_{(t)} < 2$，且因為 $W_t^T x_{n(t)}$ 一定与 $y_{n(t)}$ 異號，故 $-y_{n(t)} W_t^T x_{n(t)} \geq 0$

$$\eta(t) = \frac{|W_t^T x_{n(t)}|}{\|x_{n(t)}\|^2} + \varepsilon_{(t)}$$

$$\sum_{i=0}^{T-1} \eta(i) = \frac{|W_0^T x_{n(0)}|}{\|x_{n(0)}\|^2} + \varepsilon_{(0)} + \frac{|W_1^T x_{n(1)}|}{\|x_{n(1)}\|^2} + \varepsilon_{(1)} + \cdots + \frac{|W_{T-1}^T x_{n(T-1)}|}{\|x_{n(T-1)}\|^2} + \varepsilon_{(T-1)}$$

$$= \underbrace{\frac{|W_0^T x_{n(0)}|}{\|x_{n(0)}\|^2} + \frac{|W_1^T x_{n(1)}|}{\|x_{n(1)}\|^2} + \cdots + \frac{|W_{T-1}^T x_{n(T-1)}|}{\|x_{n(T-1)}\|^2}}_{\text{因為每一項都為非負}} \geq 0 + \underline{\varepsilon_{(0)} + \cdots + \varepsilon_{(T-1)}} \geq 1 \cdot T$$

$$\geq T$$

$$\Rightarrow \underline{\sum_{i=0}^{T-1} \eta(i) \geq T}$$

$$\sum_{i=0}^{T-1} \eta^2(i) = \boxed{\left(\frac{|W_0^T x_{n(0)}|}{\|x_{n(0)}\|^2}\right)^2 + 2\varepsilon_{(0)} \cdot \frac{|W_0^T x_{n(0)}|}{\|x_{n(0)}\|^2} + \left(\frac{|W_1^T x_{n(1)}|}{\|x_{n(1)}\|^2}\right)^2 + 2\varepsilon_{(1)} \cdot \frac{|W_1^T x_{n(1)}|}{\|x_{n(1)}\|^2} + \cdots + \left(\frac{|W_{T-1}^T x_{n(T-1)}|}{\|x_{n(T-1)}\|^2}\right)^2 + 2\varepsilon_{(T-1)} \cdot \frac{|W_0^T x_{n(T-1)}|}{\|x_{n(T-1)}\|^2}} + \boxed{\varepsilon^2_{(0)} + \varepsilon^2_{(1)} + \cdots + \varepsilon^2_{(T-1)}} \leq T \cdot 4$$

令此項為K，由於裡面每一項皆為非負，故 $\underline{0 \leq K}$

$$\Rightarrow \underline{\sum_{i=0}^{T-1} \eta^2(i) \leq K + 4T}$$

(第三題續)

代入①式

$$1 \geq \frac{\rho \cdot \sum_{i=0}^{T-1} \eta(i)}{1 \cdot R \sqrt{\sum_{i=0}^{T-1} (\eta(i))^2}} \geq \frac{\rho \cdot T}{R \cdot \sqrt{K+4T}}$$

$$\Rightarrow K+4T \geq \frac{\rho^2 T^2}{R^2}$$

$$\Rightarrow 0 \geq \frac{\rho^2}{R^2} T^2 - 4T - K$$

$$\Rightarrow 0 \geq T^2 - \frac{4R^2}{\rho^2} T - \frac{R^2}{\rho^2} K$$

$$\Rightarrow 0 \geq T^2 - \frac{4R^2}{\rho^2} T + \left(\frac{2R^2}{\rho^2}\right)^2 - \left(\frac{2R^2}{\rho^2}\right) - \frac{R^2 K}{\rho}$$

$$\Rightarrow \frac{2R^2}{\rho^2} + \frac{R^2 K}{\rho} \geq \left(T - \frac{2R^2}{\rho^2}\right)^2$$

$$\Rightarrow \frac{2R^2}{\rho^2} + \sqrt{\frac{2R^2}{\rho^2} + \frac{R^2 K}{\rho}} \geq T$$ #

收斂

P4, C

By Slide P.41 $T \le \left(\frac{R}{\rho}\right)^2$, where $R = \sqrt{\max_n \|x_n\|^2}$, $\rho = \min_n y_n w_f^T x_n$

$\boxed{求R}$, 考慮一封email會使得x_n裡有最多1 or −1,這會是擁有最多distinct word的email,而一封email最多只能有M個distinct word.

$$x_n = (\underbrace{1}_{x_0}, \underbrace{1, 1, 1, \cdots 1}_{m個distinct\ word}, 0, 0, 0, \cdots 0)$$

$$\|x_n\| = \sqrt{(m+1)\cdot 1^2} = \sqrt{m+1}, \quad R = \sqrt{m+1} = \underline{\sqrt{m+1}} \ \#$$

$\boxed{求\rho}$, 考慮 w_f 根據題意,

$$w_f = (\underbrace{-0.5}_{b}, \underbrace{1, 1, 1, \cdots 1}_{d_+}, \underbrace{-1, -1, -1, \cdots -1}_{d_-})$$

normalize

$$w_f' = w_f \times \frac{1}{\sqrt{\frac{1}{4}+d}}$$

考慮一封空白email,意即沒有任何distinct word

$$x_n = (\underbrace{1}_{x_0}, \underbrace{0, 0, \cdots 0}_{d})$$

$$w_f'^T \cdot x_n = (-0.5)\cdot\frac{1}{\sqrt{\frac{1}{4}+d}}, \quad y_n w_f'^T x_n = (-1)\cdot(-0.5)\cdot\frac{1}{\sqrt{\frac{1}{4}+d}} = \underline{\frac{\frac{1}{2}}{\sqrt{\frac{1}{4}+d}}} \ \#$$

$$\frac{R^2}{\rho^2} = \frac{m+1}{\frac{\frac{1}{4}}{\frac{1}{4}+d}} = 4(m+1)\left(d+\frac{1}{4}\right) = \underline{(m+1)(4d+1)} \ \#$$

P5, B

舉例而言,假設 $y_{n(1)}=1$, $y_{n(2)}=2$ 且 $t=2$ 之後 PLA 就是 perfect line 了.

則 $\tilde{y}_{n(1)}=-1$, $\tilde{y}_{n(2)}=1$ $\quad * y'_{(1)}=2$, $y'_{(2)}=1$

$$\begin{cases} W_{PLA} = \vec{0} + \tilde{y}_{n(1)} x_{n(1)} + \tilde{y}_{n(2)} x_{n(2)} \\ W_1^* = \vec{0} + x_{n(1)} - x_{n(2)} \\ W_2^* = \vec{0} - x_{n(1)} + x_{n(2)} \end{cases} \Rightarrow \begin{cases} W_{PLA} = -x_{n(1)} + x_{n(2)} \\ W_1^* = x_{n(1)} - x_{n(2)} \\ W_2^* = -x_{n(1)} + x_{n(2)} \end{cases}$$

$$\Rightarrow W_{PLA} = -W_1^* = W_2^*$$

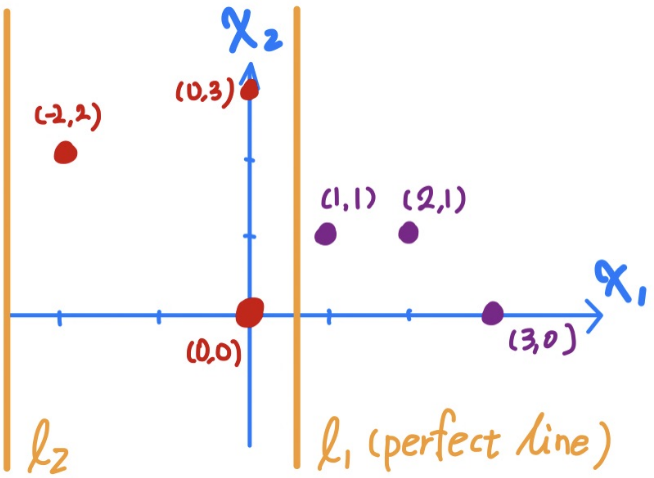
P6, D

The process isn't specifically labeled; moreover, it can be fine-tuned after learning from a simple task. Therefore, it match with the definition of self-supervised learning.

P7, C

It need to generate tag from an article, which is an application of multi-label classification. It's dataset is partially labeled, which means it's a semi-supervised learning. The algorithm will learn from all the data in a single training, which means its batch learning. The data is utf8 encoded strings which is raw features

P8, B



設 $D=\{(1,1),(2,1),(3,0)\}$

$U/D=\{(-2,2),(0,3),(0,0)\}$

$$\hat{y}=\text{sign}(w_1x_1+w_2x_2+b)$$

令 $w_1=-1$, $w_2=0$, $b=0.5$, 此為圖中 ℓ_1

$\hat{y}=\text{sign}(-w_1+0.5)$

代入 (1,1),(2,1),(3,0), $\hat{y}$ 皆為 −1

(0,3),(-2,2),(0,0), $\hat{y}$ 皆為 +1

perfect line

$E_{ots}=0$ #

令 $w_1=-1$, $w_2=0$, $b=-3$, 此為圖中 ℓ_2

$\hat{y}=\text{sign}(-w_1-3)$

代入 (1,1),(2,1),(3,0), $\hat{y}$ 皆為 −1

(0,3),(-2,2),(0,0), $\hat{y}$ 皆為 −1

U/D全部做錯

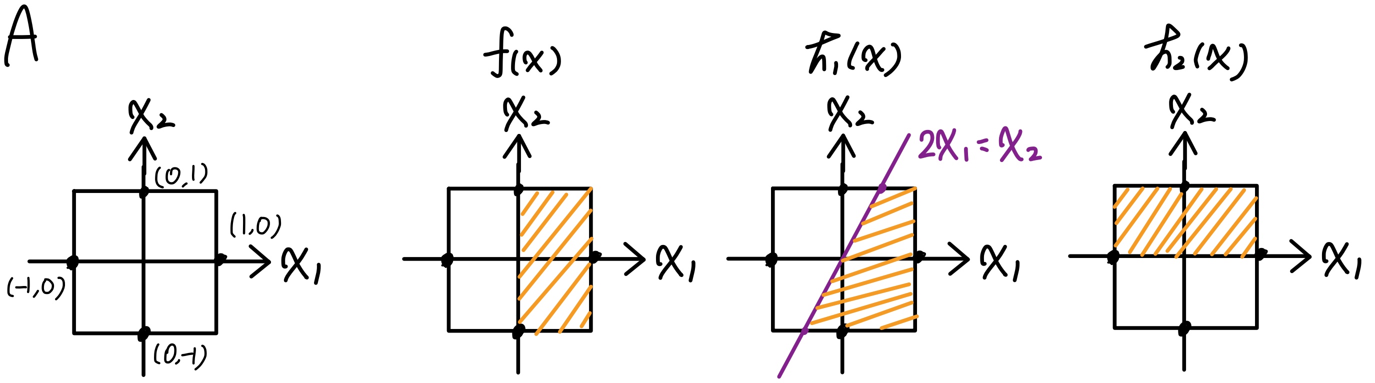
$E_{ots}=1$ #

P9. D

高斯分佈的varience為 $E[(x-\mu)^2]$，題目說該分佈為zero-mean ($\mu=0$)

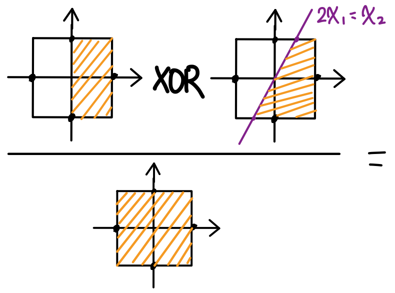
故 $\theta = E[x^2] = \frac{1}{N}\sum_{i=1}^{N} x^2 = \hat{\theta}$, where x是從高斯分佈抽樣出來的

P10. A

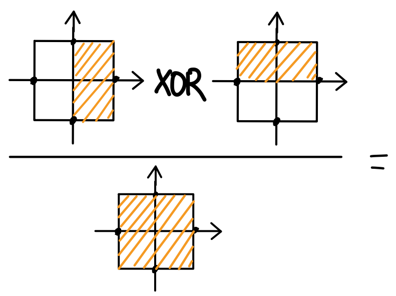


此為 $x=[x_1,x_2]\in\mathbb{R}^2$

Orange parts are sign($\vec{x}$) = 1, blank parts are sign($\vec{x}$) = -1

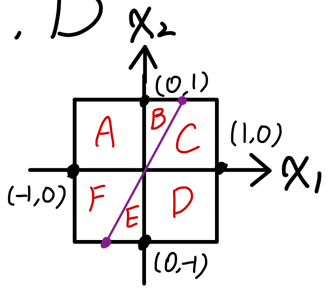
$E_{out}(h_1) =$ 

$= \frac{\frac{1}{2}}{4} = \frac{1}{8}$ #

$E_{out}(h_2) =$ 

$= \frac{2}{4} = \frac{1}{2}$ #

P11. B



將 $[-1,1]\times[-1,1]$ 切成6個区域, ABCDEF

	$[\![h_1(x)=f(x)]\!]$	$[\![h_2(x)=f(x)]\!]$	$P(x)$
A	1	-1	1/4
B	-1	1	1/16
C	1	1	3/16
D	1	-1	1/4
E	-1	1	1/16
F	1	1	3/16

$\Rightarrow$

$[\![h_1(x)=f(x)]\!]$	$[\![h_2(x)=f(x)]\!]$	$P(x)$
1	1	3/8
1	-1	4/8
-1	1	1/8
-1	-1	0

* $P(x)$ 為抽樣的sample在該区域的機率

考慮 $E_{in}(h_1)=E_{in}(h_2)=0 \Rightarrow \left(\frac{3}{8}\right)^4=\frac{81}{4096}$

$E_{in}(h_1)=E_{in}(h_2)=0.25 \Rightarrow \left(\frac{3}{8}\right)^2\times\frac{4}{8}\times\frac{1}{8}\times\frac{4!}{2!}=\frac{432}{4096}$

$E_{in}(h_1)=E_{in}(h_2)=0.5 \Rightarrow \left(\frac{4}{8}\right)^2\times\left(\frac{1}{8}\right)^2\times\frac{4!}{2!2!}=\frac{96}{4096}$

$\Rightarrow \frac{81+432+96}{4096}=\frac{609}{4096}$ #

P12, B

	1	2	3	4	5	6
A	X	O	X	O	X	O
B	O	O	X	X	X	O
C	X	X	X	X	X	O
D	X	O	O	X	O	X

* O 代表green, X 代表orange

考慮抽到的骰子的所有可能種類

一種: A, B, C, D

二種: AB, AC, AD, BC, BD, CD

三種: ABC, ABD, ACD, BCD

四種: ABCD

其中ACD, BCD, CD, ABCD 不能產生purely green number 故刪除, 所以剩下

一種: A, B, C, D $\Rightarrow 4$

二種: AB, AC, AD, BC, BD $\Rightarrow 5\times\left(\frac{5!}{4!}+\frac{5!}{3!2!}+\frac{5!}{3!2!}+\frac{5!}{4!}\right)=150$

三種: ABC, ABD $\Rightarrow 2\times\left(3\times\frac{5!}{3!}+3\times\frac{5!}{2!2!}\right)=300$

$4+150+300=454$, 此為分子

$(4)^5=4096$, 此為分母

故 answer $=\frac{454}{4096}$ #

P13. B

```
import random
import numpy as np
from numpy import linalg as LA

N = 100 # Number of trainig data

# Load training data
X = [] # Training data
Y = [] # Labels
with open('hw1_train.dat', 'r') as f:
    for i in f.readlines():
        l = [1] # X_0
        for s in i.split('\t')[:-1]:
            l.append(float(s))
        X.append(np.array(l))
        Y.append(float(i.split('\t')[-1].split('\n')[0]))

def sign(x):
    if x > 0:
        return 1
    else:
        return -1

s = 0.0
for _ in range(1000): # Do 1000 times pla
    W = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]) # weight [b, w1, w2, ..., w10]
    while True:
        is_done = True
        for _ in range(5*N): # Randomly find wrong point 5*N times
            # Randomly pick a point
            idx = random.randint(0,N-1)
            # Update weight
            if not ( sign(np.inner(W,X[idx])) == sign(Y[idx]) ):
                W = W + Y[idx]*X[idx]
                is_done = False
                break
        if is_done:
            break
    s += LA.norm(W)**2
print("Square norm of W = " + str(s/1000))
```

P14.C

```
import random
import numpy as np
from numpy import linalg as LA

N = 100 # Number of trainig data

# Load training data
X = [] # Training data
Y = [] # Labels
with open('hw1_train.dat', 'r') as f:
    for i in f.readlines():
        l = [2] # X_0
        for s in i.split('\t')[:-1]:
            l.append(float(s)*2)

        X.append(np.array(l))
        Y.append(float(i.split('\t')[-1].split('\n')[0]))

def sign(x):
    if x > 0:
        return 1
    else:
        return -1

s = 0.0
for _ in range(1000): # Do 1000 times pla
    W = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]) # weight [b, w1, w2, ..., w10]
    while True:
        is_done = True
        for _ in range(5*N): # Randomly find wrong point 5*N times
            # Randomly pick a point
            idx = random.randint(0,N-1)
            # Update weight
            if not ( sign(np.inner(W,X[idx])) == sign(Y[idx]) ):
                W = W + Y[idx]*X[idx]
                is_done = False
                break
        if is_done:
            break
    s += LA.norm(W)**2
print("Square norm of W = " + str(s/1000))
```

P15, E

```
import random
import numpy as np
from numpy import linalg as LA

N = 100 # Number of trainig data

# Load training data
X = [] # Training data
Y = [] # Labels
with open('hw1_train.dat', 'r') as f:
    for i in f.readlines():
        l = [1] # X_0
        for s in i.split('\t')[:-1]:
            l.append(float(s))

        # Use norm to normalize
        X.append(np.array(l)/LA.norm(np.array(l)))
        Y.append(float(i.split('\t')[-1].split('\n')[0]))

def sign(x):
    if x > 0:
        return 1
    else:
        return -1

s = 0.0
for _ in range(1000): # Do 1000 times pla
    W = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]) # weight [b, w1, w2, ..., w10]
    while True:
        is_done = True
        for _ in range(5*N): # Randomly find wrong point 5*N times
            # Randomly pick a point
            idx = random.randint(0,N-1)
            # Update weight
            if not ( sign(np.inner(W,X[idx])) == sign(Y[idx]) ):
                W = W + Y[idx]*X[idx]
                is_done = False
                break
        if is_done:
            break
    s += LA.norm(W)**2
print("Square norm of W = " + str(s/1000))
```

P16. A

```
import random
import numpy as np
from numpy import linalg as LA

N = 100 # Number of trainig data

# Load training data
X = [] # Training data
Y = [] # Labels
with open('hw1_train.dat', 'r') as f:
    for i in f.readlines():
        l = [0] # X_0
        for s in i.split('\t')[:-1]:
            l.append(float(s))

        X.append(np.array(l))
        Y.append(float(i.split('\t')[-1].split('\n')[0]))

def sign(x):
    if x > 0:
        return 1
    else:
        return -1

s = 0.0
for _ in range(1000): # Do 1000 times pla
    W = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]) # weight [b, w1, w2, ..., w10]
    while True:
        is_done = True
        for _ in range(5*N): # Randomly find wrong point 5*N times
            # Randomly pick a point
            idx = random.randint(0,N-1)
            # Update weight
            if not ( sign(np.inner(W,X[idx])) == sign(Y[idx]) ):
                W = W + Y[idx]*X[idx]
                is_done = False
                break
        if is_done:
            break
    s += LA.norm(W)**2
print("Square norm of W = " + str(s/1000))
```