HTML Homework 3

1. B

since we want to do a binary classifier, so we have $\binom{K}{2}$ choices, each choices with $2 \cdot \dfrac{N}{K}$ data size

the CPU time = $\binom{K}{2} \cdot a \cdot \dfrac{2N}{K} = \dfrac{k(k-1)}{2} \dfrac{2N \cdot a}{K} = a(K-1)(N)$

2. E

we know that $y = w^T x$, so if we can found x^{-1} then we must can get our w for all our inputs.

$\because w^T = yx^{-1}$

however, now we assume our w is with six parameters $(1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$, so we need to transform our x to this form.

let the new transformation of x be z

$$\Rightarrow z = \begin{bmatrix} 1 & 2 & 0 & 0 & 4 & 0 \\ 1 & 0 & 2 & 0 & 0 & 4 \\ 1 & -2 & 0 & 0 & 4 & 0 \\ 1 & 0 & -2 & 0 & 0 & 4 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

then we use the invert matrix calculator, we can found that z^{-1} exists

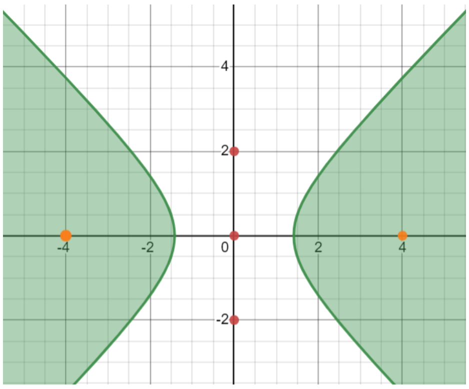
$$\Rightarrow z^{-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{1}{4} & 0 & \frac{-1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & \frac{-1}{4} & 0 & 0 \\ \frac{-3}{8} & \frac{-3}{8} & \frac{1}{8} & \frac{1}{8} & \frac{-1}{2} & 1 \\ \frac{1}{8} & 0 & \frac{1}{8} & 0 & \frac{-1}{4} & 0 \\ 0 & \frac{1}{8} & 0 & \frac{1}{8} & \frac{-1}{4} & 0 \end{bmatrix}$$

so that means we must can shatter all six inputs no matter what y each input has.

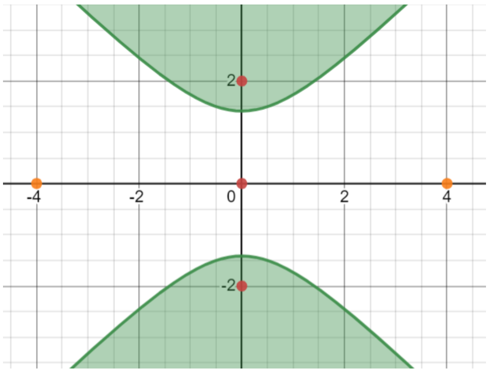
3. C

for these five inputs, I will show the ones with positive y in yellow, and the others in red.

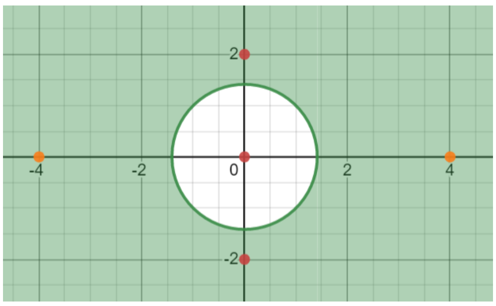
for $w1$, we can see that all 5 inputs are shattered correctly.



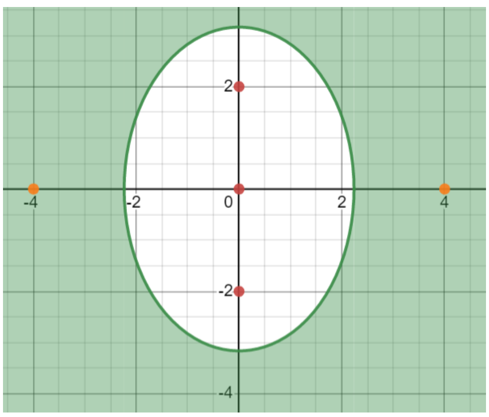
for w_2, we can see that the all 5 inputs are shattered incorrectly.



for w_3, we can see that x_4 and x_5 is shattered incorrectly.



for w_4, we can see that all 5 inputs are shattered correctly.



Then we can know that only w_1 and w_4 can be a linear classifier for these inputs.

4. B

Although we transform the input x into Γx, we still get the same y label.

So both of the w we found should match this condition.

$\Rightarrow y = w_{lin}^T x = \widetilde{w}^T \Gamma x$

$\Rightarrow x^T w_{lin} = x^T \Gamma^T \widetilde{w} \Rightarrow w_{lin} = \Gamma^T \widetilde{w}$

for the E_{in},

because the w is just a linear transformation, so it shouldn't affect the wrong decision

then the $E_{in}(w_{lin}) = E_{in}(\widetilde{w})$

5. B

The description has mentioned that H_k has the growth function $2N$

$\Rightarrow$ the growth function for $\bigcup_{k=1}^{d} H_k$ is at most $d \cdot 2N$

If we let the d_{vc} has the upper bound N, that is $d_{vc} \geq N$

then based on the growth function's definition, we can know that $d \cdot 2N \geq 2^N$

$\Rightarrow N \leq log_2 d + log_2 2 + log_2 N \leq log_2 d + log_2 2 + \frac{N}{2}$

$\Rightarrow N \leq 2(log_2 d + 1)$

6. C

for (c),

if $g(x_n) = 1$, which means that $x = x_n$

but when we scale the x_n to $2x_n$, then $x \neq x_n$ which will make our $z_n = 0$

in this case, $g(2x_n) = 0 \neq 1 = y_n$

7. E

since we want to minimize E_{aug} in L1-regularized, and we know that $N = 3$ and $\lambda = 3$ from the description.

Also, when we point out the three inputs, we can see that the optimal line must be left up right down, with a positive w_0 and a positive w_1

so we need to minimize this function:

$$\begin{aligned} E_{aug}(w) &= E_{in}(w) + |w| \\ &= \frac{1}{3}((w_0 + 2w_1 - 1)^2 + (w_0 + 3w_1)^2 + (w_0 - 2w_1 - 2)^2) + |w_0| + |w_1| \\ &= \frac{1}{3}((w_0 + 2w_1 - 1)^2 + (w_0 + 3w_1)^2 + (w_0 - 2w_1 - 2)^2) + w_0 - w_1 \end{aligned}$$

$$\frac{\partial E_{aug}}{w_0} = \frac{1}{3}(2(w_0 + 2w_1 - 1) + 2(w_0 + 3w_1) + 2(w_0 - 2w_1 - 2)) + 1$$

$$\Rightarrow 6w_0 + 6w_1 - 3 = 0$$

$$\frac{\partial E_{aug}}{w_1} = \frac{1}{3}(4(w_0 + 2w_1 - 1) + 6(w_0 + 3w_1) - 4(w_0 - 2w_1 - 2)) - 1$$

$$\Rightarrow 6w_0 + 34w_1 + 1 = 0$$

$$\Rightarrow w_0 = \frac{9}{14}, w_1 = \frac{-1}{7}$$

then we can calculate our E_{aug}

$$\begin{aligned} E_{aug}(w) &= \frac{1}{3}((\frac{9}{14} + 2 \cdot \frac{-1}{7} - 1)^2 + (\frac{9}{14} + 3 \cdot \frac{-1}{7})^2 + (\frac{9}{14} - 2 \cdot \frac{-1}{7} - 2)^2) + \frac{9}{14} - \frac{-1}{7} \\ &= \frac{105}{196} + \frac{126}{196} + \frac{28}{196} = \frac{259}{195} = 1.32 \end{aligned}$$

8. B

since we want to minimize E_{aug} in L2-regularized, then we want $\nabla E_{aug}(w) = \nabla E_{in}(w) + \frac{2\lambda}{N}|w| = 0$

then our optimal solution for $w = (Z^T Z + \lambda I)^{-1} Z^T y$

$$w = (Z^T Z + \lambda I)^{-1} Z^T y$$

$$= \left(\begin{bmatrix} 1 & 1 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 1 & -2 \end{bmatrix} + \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} 9 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} 2+\lambda & 0 \\ 0 & 8+\lambda \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} 9 \\ -1 \end{bmatrix}$$

$$= \frac{1}{(2+\lambda)(8+\lambda)} \begin{bmatrix} 8+\lambda & 8+\lambda \\ 4+2\lambda & -4-2\lambda \end{bmatrix} \begin{bmatrix} 9 \\ -1 \end{bmatrix}$$

$$= \frac{1}{(2+\lambda)(8+\lambda)} \begin{bmatrix} 64+8\lambda \\ 40+20\lambda \end{bmatrix}$$

from the above steps, we can get our $w = [\frac{64+8\lambda}{(2+\lambda)(8+\lambda)}, \frac{40+20\lambda}{(2+\lambda)(8+\lambda)}]$

with the test example $(x = 1, y = 4)$, we want to make $4 = w_0 + w_1$

so we solve the equation: $\frac{64+8\lambda}{(2+\lambda)(8+\lambda)} + \frac{40+20\lambda}{(2+\lambda)(8+\lambda)} = 4$

$\Rightarrow 104 + 28\lambda = 4\lambda^2 + 40\lambda + 64$

$\Rightarrow \lambda = -5$ or 2

we will choose $\lambda = 2$ because there is a condition that $\lambda > 0$

9. B

we can know that the $\nabla E_{aug}(w) = \nabla E_{in}(w) + \frac{2\lambda}{N}w$

$\Rightarrow w_{t+1} = w_t - \eta \nabla E_{aug}(w_t) = w_t - \eta(\nabla E_{in}(w_t) + \frac{2\lambda}{N}w_t) = (1 - \eta\frac{2\lambda}{N})w_t - \eta\nabla E_{in}(w_t)$

then we can get that the $\alpha = (1 - \eta\frac{2\lambda}{N})$

10. B

based on the lecture slide, we can know the solution of linear regression

$w_{reg} = (X^TX + \lambda I)^{-1}X^TY$

for the K virtual examples, we want to solve

$\min_{w\in\mathbb{R}^{d+1}} \frac{1}{N+K}(\sum_{n=1}^{N}(y_n - w^Tx_n)^2 + \sum_{k=1}^{K}(\tilde{y}_k - w^T\tilde{x}_k)^2)$

we first transform it into the matrix form

$\Rightarrow \min_{w\in\mathbb{R}^{d+1}} \frac{1}{N+K}((Y - w^TX)^2 + (\widetilde{Y} - w^T\widetilde{X})^2)$

if we want to have the minimum, then we want its $\nabla = 0$

$\Rightarrow \nabla(\frac{1}{N+K}((Y - w^TX)^2 + (\widetilde{Y} - w^T\widetilde{X})^2)) = \frac{1}{N+K}(2(Y - w^TX)(-X) + 2(\widetilde{Y} - w^T\widetilde{X})(-\widetilde{X})$

$\Rightarrow \frac{1}{N+K}(-2X^TY + 2w^TX^TX - 2\widetilde{X}^T\widetilde{Y} + 2w^T\widetilde{X}^T\widetilde{X}) = 0$

$\Rightarrow w = (X^TX + \widetilde{X}^T\widetilde{X})^{-1}(X^TY + \widetilde{X}\widetilde{Y})$

we want $(X^TX + \widetilde{X}^T\widetilde{X})^{-1}(X^TY + \widetilde{X}\widetilde{Y}) = (X^TX + \lambda I)^{-1}X^TY$

so $\widetilde{X}^T\widetilde{X} = \lambda I_{d+1}$ and $\widetilde{X}\widetilde{Y} = 0$

$\Rightarrow \widetilde{X} = \sqrt{\lambda}I_{d+1}$ and $\widetilde{Y} = 0$

11. C

let H_{ij} be the element of the X_h^TX

$$\begin{aligned}\mathbb{E}(H_{ij}) &= \mathbb{E}(\sum_{k=1}^{N}(x_k \cdot x_k + x_k \cdot x_k + 2 \cdot x_k \cdot \epsilon + \epsilon^2)) \\ &= \mathbb{E}(\sum_{k=1}^{N} 2x_k^2) + 2\mathbb{E}(\sum_{k=1}^{N} x_k) \cdot \mathbb{E}(\epsilon) + N \cdot \mathbb{E}(\epsilon^2) \\ &= \mathbb{E}(\sum_{k=1}^{N} 2x_k^2) + N \cdot \frac{r^2}{3}\end{aligned}$$

$\because \mathbb{E}(\epsilon) = \int_{-r}^{r} \frac{1}{2r}xdx = \frac{1}{2r} \cdot (\frac{r^2}{2} - \frac{r^2}{2}) = 0$ and $\mathbb{E}(\epsilon^2) = \int_{-r}^{r} \frac{1}{2r}x^2dx = \frac{1}{2r} \cdot (\frac{r^3}{3} + \frac{r^3}{3}) = \frac{r^3}{3}$

then we could know that $\mathbb{E}(X_h^TX_h)$

$\mathbb{E}(X_h^TX_h) = 2X^TX + \frac{Nr^3}{3}I_{d+1}$

12. B

the optimal solution of $\min_{y\in\mathbb{R}} \frac{1}{N}\sum_{n=1}^{N}(y-y_n)^2+\frac{\lambda}{N}\Omega(y)$ is that we want its ∇ = 0

$$\Rightarrow \nabla(\frac{1}{N}\sum_{n=1}^{N}(y-y_n)^2+\frac{\lambda}{N}\Omega(y))=\frac{1}{N}\sum_{n=1}^{N}(2y-2y_n)+\frac{\lambda}{N}\Omega'(y)=\frac{2yN}{N}-\frac{\sum_{n=1}^{N}2y_n}{N}+\frac{\lambda}{N}\Omega'(y)=0$$

Also, we know that $y=\frac{(\sum_{n=1}^{N}y_n)+K}{N+2K}$

Then from the four choices, we can put them into the equation to see whether we can come out with the same solution for the optimal y

(a)

$$\Omega'(y)=\frac{2K}{\lambda}(2y+1)$$

$$\Rightarrow \frac{2yN}{N}-\frac{\sum_{n=1}^{N}2y_n}{N}+\frac{\lambda}{N}\frac{2K}{\lambda}(2y+1)=0$$

$$\Rightarrow y=\frac{\sum_{n=1}^{N}y_n-K}{N+2K}\neq\frac{(\sum_{n=1}^{N}y_n)+K}{N+2K}$$

(b)

$$\Omega'(y)=\frac{2K}{\lambda}(2y-1)$$

$$\Rightarrow \frac{2yN}{N}-\frac{\sum_{n=1}^{N}2y_n}{N}+\frac{\lambda}{N}\frac{2K}{\lambda}(2y-1)=0$$

$$\Rightarrow y=\frac{\sum_{n=1}^{N}y_n+K}{N+2K}=\frac{(\sum_{n=1}^{N}y_n)+K}{N+2K}$$

$$\Rightarrow \Omega(y)=\frac{2K}{\lambda}(y-0.5)^2$$

(c)

$$\Omega'(y)=\frac{K}{\lambda}(2y+1)$$

$$\Rightarrow \frac{2yN}{N}-\frac{\sum_{n=1}^{N}2y_n}{N}+\frac{\lambda}{N}\frac{K}{\lambda}(2y+1)=0$$

$$\Rightarrow y=\frac{\sum_{n=1}^{N}y_n-K}{N+K}\neq\frac{(\sum_{n=1}^{N}y_n)+K}{N+2K}$$

(d)

$$\Omega'(y)=\frac{K}{\lambda}(2y-1)$$

$$\Rightarrow \frac{2yN}{N}-\frac{\sum_{n=1}^{N}2y_n}{N}+\frac{\lambda}{N}\frac{K}{\lambda}(2y-1)=0$$

$$\Rightarrow y=\frac{\sum_{n=1}^{N}y_n+K}{N+K}\neq\frac{(\sum_{n=1}^{N}y_n)+K}{N+2K}$$

13. C

```python
import numpy as np
from numpy import loadtxt
import statistics
import math

file = open('hw3train.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data.append(numbers)

x0 = 1
for row in data:
    row.insert(0, x0)

x = np.array(data)
y = x[:, -1]
x = x[:, :-1]
xt = x.transpose()

inv = np.matmul(xt, x)
inv = np.linalg.inv(inv)

wlin = np.matmul(inv, xt)
wlin = np.matmul(wlin, y)

print(wlin)
Esqr = float(0)
for i in range(len(x)):
    h = float(np.matmul(wlin.transpose(), x[i]))
    err = float(h - y[i])
    Esqr += float(pow(err, 2))
Esqr /= float(len(x))
print(Esqr)
```

I get the $E_{in}^{sqr} = 0.7922347761105571$

Also, we can get the
$w_{lin} = [0.29070963, -0.04988084, 0.04893561, -0.08623605, -0.06658103, 0.10689752, -0.12356574, 0.09486241, 0.26696655, -0.15660245, -0.06382855]$

14. D

```
import numpy as np
from numpy import loadtxt
import random
from random import randint
import statistics
import math

file = open('hw3train.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data.append(numbers)

x0 = 1
for row in data:
    row.insert(0, x0)

x = np.array(data)
y = x[:, -1]
x = x[:, :-1]

n = 0.001
Eavg = float(0)

for i in range(1000):
    wlin = np.zeros(len(x[0]))
    Esqr = float(0)
    for j in range(800):
        m = randint(0, len(x)-1)
        er = float(np.matmul(wlin.transpose(), x[m]))
        er -= y[m]
        er *= -2
        er *= n
        wlin += er*x[m]
    for j in range(len(x)):
        h = float(np.matmul(wlin.transpose(), x[j]))
        Esqr += float(pow((h-y[j]),2))
    Esqr /= float(len(x))
    Eavg += Esqr

Eavg /= float(1000)
print(Eavg)
```

for SGD in linear regression, we can know that the
$-\nabla err = -\nabla((w^T x_n - y_n)^2) = -2(w^T x_n - y_n)(x_n)$

I get the $E_{in}^{sqr} = 0.8235212901376975$

15. C

```
import numpy as np
from numpy import loadtxt
import random
from random import randint
import statistics
import math

file = open('hw3train.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data.append(numbers)

x0 = 1
for row in data:
    row.insert(0, x0)

x = np.array(data)
y = x[:, -1]
x = x[:, :-1]

n = 0.001
Eavg = float(0)

for i in range(1000):
    wlin = np.zeros(len(x[0]))
    Ece = float(0)
    for j in range(800):
        m = randint(0, len(x)-1)
        er = np.matmul(wlin.transpose(), x[m])
        er = y[m]*er
        theta = float(1/(1 + pow(math.e, er)))
        final = n*theta
        final *= y[m]*x[m]
        wlin += final
    for j in range(len(x)):
        h = float(np.matmul(wlin.transpose(), x[j]))
        h *= -1
        h *= y[j]
        Ece += np.log(1+pow(math.e, h))
    Ece /= float(len(x))
    Eavg += Ece

Eavg /= float(1000)
print(Eavg)
```

I get the $E_{in}^{ce} = 0.6571606836285517$

16. A

```
import numpy as np
from numpy import loadtxt
import random
from random import randint
import statistics
import math

file = open('hw3train.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data.append(numbers)

x0 = 1
for row in data:
    row.insert(0, x0)

x = np.array(data)
y = x[:, -1]
x = x[:, :-1]

n = 0.001
Eavg = float(0)

for i in range(1000):
    w0 = np.array([0.29070963, -0.04988084, 0.04893561, -0.08623605, -0.06658103,
0.10689752, -0.12356574,  0.09486241, 0.26696655, -0.15660245, -0.06382855])
    #print(w0)
    Ece = float(0)
    for j in range(800):
        m = randint(0, len(x)-1)
        er = np.matmul(w0.transpose(), x[m])
        er = y[m]*er
        theta = float(1/(1 + pow(math.e, er)))
        final = n*theta
        final *= y[m]*x[m]
        w0 += final
    for j in range(len(x)):
        h = float(np.matmul(w0.transpose(), x[j]))
        h *= -1
        h *= y[j]
        Ece += np.log(1+pow(math.e, h))
    Ece /= float(len(x))
    Eavg += Ece

Eavg /= float(1000)
print(Eavg)
```

we can know that $w_{lin} = [0.29070963, -0.04988084, 0.04893561, -0.08623605, -0.06658103, 0.10689752, -0.12356574, 0.09486241, 0.26696655, -0.15660245, -0.06382855]$

from Q13, so we can just copy the result and assign it to w_0

then we get the $E_{in}^{ce} = 0.6051474036432736$

17. A

```
import numpy as np
from numpy import loadtxt
import random
from random import randint
import statistics
import math

file = open('hw3train.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data.append(numbers)

file2 = open('hw3test.txt', 'r')
data2 = []
for line in file2:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data2.append(numbers)

x0 = 1
for row in data:
    row.insert(0, x0)

for row in data2:
    row.insert(0, x0)

x = np.array(data)
y = x[:, -1]
x = x[:, :-1]

xtest = np.array(data2)
ytest = xtest[:, -1]
xtest = xtest[:, :-1]

n = 0.001
Eavg = float(0)

for i in range(1000):
    w0 = np.array([0.29070963, -0.04988084, 0.04893561, -0.08623605, -0.06658103,
0.10689752, -0.12356574,  0.09486241, 0.26696655, -0.15660245, -0.06382855])
    Ein = float(0)
    Eout = float(0)
    for j in range(800):
        m = randint(0, len(x)-1)
        er = np.matmul(w0.transpose(), x[m])
        er = y[m]*er
        theta = float(1/(1 + pow(math.e, er)))
        final = n*theta
        final *= y[m]*x[m]
        w0 += final
    for j in range(len(x)):
        if(y[j]*(np.matmul(w0.transpose(), x[j])) <= 0):
            Ein += 1
```

```
    Ein /= float(len(x))
    for j in range(len(xtest)):
        if(ytest[j]*(np.matmul(w0.transpose(), xtest[j])) <= 0):
            Eout += 1
    Eout /= float(len(xtest))
    Eavg += abs(Ein - Eout)

Eavg /= float(1000)
print(Eavg)
```

same as Q17, we just assign w_0 to the result we get from Q13, and do logistic regression. Then calculate the $0/1$ error.

I get the $|E_{in}^{0/1} - E_{out}^{0/1}| = 0.030575000000000054$

18. B

```
import numpy as np
from numpy import loadtxt
import random
from random import randint
import statistics
import math

file = open('hw3train.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data.append(numbers)

file2 = open('hw3test.txt', 'r')
data2 = []
for line in file2:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data2.append(numbers)

x0 = 1
for row in data:
    row.insert(0, x0)

for row in data2:
    row.insert(0, x0)

x = np.array(data)
y = x[:, -1]
x = x[:, :-1]
xt = x.transpose()

inv = np.matmul(xt, x)
inv = np.linalg.inv(inv)

wlin = np.matmul(inv, xt)
wlin = np.matmul(wlin, y)

xtest = np.array(data2)
ytest = xtest[:, -1]
xtest = xtest[:, :-1]

n = 0.001
Eavg = float(0)

Ein = float(0)
Eout = float(0)

for j in range(len(x)):
    if(y[j]*(np.matmul(wlin.transpose(), x[j])) <= 0):
        Ein += 1
Ein /= float(len(x))

for j in range(len(xtest)):
```

```
    if(ytest[j]*(np.matmul(wlin.transpose(), xtest[j])) <= 0):
        Eout += 1
Eout /= float(len(xtest))

Eavg = abs(Ein - Eout)
print(Eavg)
```

I get the $|E_{in}^{0/1} - E_{out}^{0/1}| = 0.040000000000000036$

19. C

```python
import numpy as np
from numpy import loadtxt
import random
from random import randint
import statistics
import math

file = open('hw3train.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data.append(numbers)

file2 = open('hw3test.txt', 'r')
data2 = []
for line in file2:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data2.append(numbers)

x0 = 1
for row in data:
    row.insert(0, x0)
    for j in range(1, 11):
        row.insert(len(row)-1, pow(row[j], 2))

for row in data2:
    row.insert(0, x0)
    for j in range(1, 11):
        row.insert(len(row)-1, pow(row[j], 2))

x = np.array(data)
y = x[:, -1]
x = x[:, :-1]

xt = x.transpose()
inv = np.matmul(xt, x)
inv = np.linalg.inv(inv)

wlin = np.matmul(inv, xt)
wlin = np.matmul(wlin, y)

xtest = np.array(data2)
ytest = xtest[:, -1]
xtest = xtest[:, :-1]

Eavg = float(0)
Ein = float(0)
Eout = float(0)

for j in range(len(x)):
    if(y[j]*(np.matmul(wlin.transpose(), x[j])) <= 0):
        Ein += 1
Ein /= float(len(x))
```

```
for j in range(len(xtest)):
    if(ytest[j]*(np.matmul(wlin.transpose(), xtest[j])) <= 0):
        Eout += 1
Eout /= float(len(xtest))

Eavg = abs(Ein - Eout)
print(Eavg)
```

I get the $|E_{in}^{0/1} - E_{out}^{0/1}| = 0.08249999999999999$

20. D

```
import numpy as np
from numpy import loadtxt
import random
from random import randint
import statistics
import math

file = open('hw3train.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data.append(numbers)

file2 = open('hw3test.txt', 'r')
data2 = []
for line in file2:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data2.append(numbers)

x0 = 1
for row in data:
    row.insert(0, x0)
    for k in range(2, 9):
        for j in range(1, 11):
            row.insert(len(row)-1, pow(row[j], k))

for row in data2:
    row.insert(0, x0)
    for k in range(2, 9):
        for j in range(1, 11):
            row.insert(len(row)-1, pow(row[j], k))

x = np.array(data)
y = x[:, -1]
x = x[:, :-1]

xt = x.transpose()
inv = np.matmul(xt, x)
inv = np.linalg.inv(inv)

wlin = np.matmul(inv, xt)
wlin = np.matmul(wlin, y)

xtest = np.array(data2)
ytest = xtest[:, -1]
xtest = xtest[:, :-1]

Eavg = float(0)

Ein = float(0)
Eout = float(0)

for j in range(len(x)):
```

```
    if(y[j]*(np.matmul(wlin.transpose(), x[j])) <= 0):
        Ein += 1
Ein /= float(len(x))

for j in range(len(xtest)):
    if(ytest[j]*(np.matmul(wlin.transpose(), xtest[j])) <= 0):
        Eout += 1
Eout /= float(len(xtest))

Eavg = abs(Ein - Eout)
print(Eavg)
```

I get the $|E_{in}^{0/1} - E_{out}^{0/1}| = 0.415$