R10521516 廖浚評

1.(D)

1. Consider a noisy target $y = \mathbf{w}_f^T \mathbf{x} + \epsilon$, where $\mathbf{x} \in \mathbb{R}^{d+1}$ (including the added coordinate $x_0 = 1$), $y \in \mathbb{R}$, $\mathbf{w}_f \in \mathbb{R}^{d+1}$ is an unknown vector, and ϵ is an i.i.d. noise term with zero mean and σ^2 variance. Assume that we run linear regression on a training data set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ generated i.i.d. from some $P(\mathbf{x})$ and the noise process above, and obtain the weight vector $\mathbf{w}_{\text{lin}}$. As briefly discussed in Lecture 5, it can be shown that the expected in-sample error $E_{\text{in}}(\mathbf{w}_{\text{lin}})$ with respect to $\mathcal{D}$ is given by:

$$\mathbb{E}_{\mathcal{D}}\left[E_{\text{in}}(\mathbf{w}_{\text{lin}})\right] = \sigma^2 \left(1 - \frac{d+1}{N}\right).$$

For $\sigma = 0.1$ and $d = 19$, what is the smallest number of examples N such that $\mathbb{E}_{\mathcal{D}}\left[E_{\text{in}}(\mathbf{w}_{\text{lin}})\right]$ is no less than 0.005? Choose the correct answer; explain your answer.

[a] 25

[b] 30

[c] 35

[d] 40

[e] 45

$0.005 \le (0.1)^2 \times \left(1 - \left(\frac{19+1}{N}\right)\right)$

$0.5 \le 1 - \frac{20}{N}$

$\frac{20}{N} \le 0.5$

1 of 7

$40 \le N$

2.(C)

$E(\text{out})g = \frac{1}{5} - \frac{2}{3}w_0 - \frac{1}{2}w_1 + w_0^2 + w_0 w_1 + \frac{1}{3}w_1^2$

$\frac{\partial E(g)}{\partial w_1} = -\frac{1}{2} + w_0 + \frac{2}{3}w_1 = 0$

$\frac{\partial E(g)}{\partial w_0} = -\frac{2}{3} + 2w_0 + w_1 = 0$

$\begin{cases} w_0 = -\frac{1}{6} \\ w_1 = 1 \end{cases}$

$L(w_0, w_1) = \int_0^1 (x^2 - (w_0 + w_1 x))^2 dx$

$= \frac{1}{5}x^5\Big|_0^1 - \frac{2}{3}w_0 x^3\Big|_0^1 - \frac{2}{4}w_1 x^4\Big|_0^1 + w_0^2 x\Big|_0^1$

$= \frac{1}{5} - \frac{2}{3}w_0 - \frac{1}{2}w_1 + w_0^2 + w_0 w_1 + \frac{1}{3}w_1^2$

3.(E)

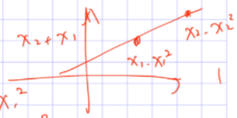
$\int_0^1 \left(g(x_1 - x_2)(x) - x^2\right)^2 dx = \frac{1}{5} - \frac{2}{3}w_0 - \frac{1}{2}w_1 + w_0^2 + w_0 w_1 + \frac{1}{3}w_1^2$

$h(x) = w_0 + w_1 x$

$w_1 =$ 斜率　$w_0 =$ 截距

$w_1 = m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{x_2^2 - x_1^2}{x_2 - x_1} = x_2 + x_1$

$w_0 = -x_1 x_2$



$(x_2 + x_1)x_1 + w_0 = x_1^2$

$x_2 x_1 + x_1^2 + w_0 = x_1^2$

$w_0 = -x_2 x_1$

$\rightarrow \frac{1}{5} - \frac{2}{3}(-x_1 x_2) - \frac{1}{2}(x_2 + x_1)$

$+ (-x_1 x_2)^2 + (-x_1 x_2)(x_2 + x_1)$

$+ \frac{1}{3}(x_2 + x_1)^2$

$\Rightarrow \frac{1}{5} - \frac{2}{3}\left(-\frac{1}{4}\right) - \frac{1}{2}(1)$

$+ \left(\frac{1}{9}\right) + \left(-\frac{1}{6}\right) + \left(-\frac{1}{6}\right) + \frac{7}{18}$

$= \frac{1}{5} + \frac{1}{6} - \frac{1}{2} + \frac{1}{9} - \frac{1}{3} + \frac{7}{18}$

$= \frac{1}{5} + \frac{1}{6} - \frac{1}{3} + \left(\frac{1}{9} + \frac{7}{18} - \frac{1}{2}\right)$

$= \frac{11 - 10}{30} + 0$

$= \frac{1}{30}$

$E(x_1 x_2) = \int_0^1\int_0^1 x_1 x_2 = \frac{1}{4}x_1^2 x_2^2\Big|_0^1 = \frac{1}{4}$

$E(-x_1 x_2^2) = \int_0^1\int_0^1 -x_1 x_2^2 = -\int_0^1 \frac{1}{2}x_1^2 x_2^2 = -\int \frac{1}{6}x_1^2 x_2^3 = -\frac{1}{6}$

$E(-x_1^2 x_2) = -\frac{1}{6}$

$E(x_1 + x_2) = \int\int x_1 + x_2 = \int \frac{1}{2}x_1^2 + x_1 x_2 = \frac{1}{2}x_1^2 x_2 + \frac{1}{2}x_1 x_2^2 = 1$

$E(x_1 x_2)^2 = E(x_1^2 x_2^2) = \int\int x_1^2 x_2^2 = \frac{1}{3}x_1^3 \frac{1}{3}x_2^3 = \frac{1}{9}$

$E(x_2 + x_1)^2 = E(x_2^2 + 2x_1 x_2 + x_1^2)$

$\frac{1}{3} + \frac{1}{2} + \frac{1}{3} = \frac{7}{6}$

$\int\int x_2^2 = \int\int x_2^2 x_1 = \int \frac{1}{3}x_2^3 x_1$

頁次: 3 / 11

4.(B)

4. In class, we introduced our version of the cross-entropy error function

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} - \ln \theta(y_n \mathbf{w}^T \mathbf{x}_n).$$

$\iint E_{out}\, dx_1 dx_2$

based on the definition of $y_n \in \{-1, +1\}$. If we transform y_n to $y'_n \in \{0, 1\}$ by $y'_n = \frac{y_n+1}{2}$, which of the following error function is equivalent to E_{in} above? Choose the correct answer; explain your answer.

B

[a] $\frac{1}{N}\sum_{n=1}^{N}\Big(y'_n \ln \theta(\mathbf{w}^T\mathbf{x}_n) + (1 - y'_n)\ln(\theta(-\mathbf{w}^T\mathbf{x}_n))\Big)$

[b] $\frac{1}{N}\sum_{n=1}^{N}\Big(y'_n \ln \theta(-\mathbf{w}^T\mathbf{x}_n) + (1 - y'_n)\ln(\theta(\mathbf{w}^T\mathbf{x}_n))\Big)$ ✓

[c] $\frac{1}{N}\sum_{n=1}^{N}\Big(y'_n \ln \theta(\mathbf{w}^T\mathbf{x}_n) - (1 - y'_n)\ln(\theta(-\mathbf{w}^T\mathbf{x}_n))\Big)$

[d] $\frac{1}{N}\sum_{n=1}^{N}\Big(y'_n \ln \theta(-\mathbf{w}^T\mathbf{x}_n) - (1 - y'_n)\ln(\theta(\mathbf{w}^T\mathbf{x}_n))\Big)$

[e] none of the other choices

when $y_n = -$

when $y_n' = 0$ 時

$1 - y_n'$

we got 後式

when $y_n' = 1$ 時

$y_n = 1$

前式

$\min -\ln\theta(y_n w^T x_n)$

$\Rightarrow \min -\theta(y_n w^T x_n)$

$\Rightarrow \min -(y_n w^T x_n)$

$\Rightarrow \min -y_n \ln\theta(w^T x_n)$

$\rightarrow y_n = -1 \rightarrow \ln\theta(w^T x_n)\quad y_n' = 0$

$y_n = +1 \rightarrow \ln\theta(w^T x_n)\quad y_n' = 1$

5.(E)

$P(\mu \le \mu) \ge 1-\delta$

$P(|\nu-\mu| > \epsilon) \le \delta$

$Pr(|\nu-\mu|>\epsilon) \le 2\exp(-2\epsilon^2 N)$

$2\exp(-2\epsilon^2 N) \le \delta$

$-2\epsilon^2 N \le \ln\frac{\delta}{2}$

$-\epsilon^2 \le \frac{1}{2N}\ln\frac{\delta}{2}$

$\epsilon = \sqrt{-\frac{1}{2N}\ln\frac{\delta}{2}}$

$\epsilon \le \sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}$

$\Rightarrow \mu \le \nu + \sqrt{\frac{1}{2N}\ln 2/\delta}$ Yes

upper bound

by hoeffding's inequality

$\text{Likelihood}(\hat{\mu}) = \prod P(x_i)P(y|\hat{\mu})$

$P(1|\hat{\mu}) = \hat{\mu}$

$P(0|\hat{\mu}) = 1-\hat{\mu}$

$\Rightarrow \max \sum \ln(P(y_i|\hat{\theta}))$

if $y=1$ 有a個 $y=0$ 有b個

$$\Rightarrow \frac{d\sum\ln(P(y_i|\hat{\theta}))}{d\hat{\theta}} = \frac{d(a\ln\hat{\theta} + b\ln(1-\hat{\theta}))}{d\hat{\theta}}$$

$$\Rightarrow \frac{a}{\hat{\theta}} - \frac{b}{1-\hat{\theta}} = 0$$

$$a(1-\hat{\theta}) - b\hat{\theta} = 0$$

$$\hat{\theta} = \frac{a}{a+b} = \nu$$ Yes

$$\frac{dE^{sqr}}{d\hat{y}} = \frac{\frac{1}{N}\sum(\hat{y}^2 - 2\hat{y}y_n + y_n^2)}{d\hat{y}}$$

$$\Rightarrow 2\hat{y} - \frac{2}{N}\sum y_n = 0$$

$$\Rightarrow \hat{y} = \frac{1}{N}\sum y_n \Rightarrow \nu$$

Yes

$$\frac{dE^{ce}}{d\hat{y}} = \frac{-\frac{1}{N}\sum[y_n\ln\hat{y} + (1-y_n)\ln(1-\hat{y})]}{d\hat{y}}$$

$$= -\left[\frac{\frac{1}{N}\sum y_n}{\hat{y}} + \frac{-1}{1-\hat{y}} + \frac{\frac{1}{N}\sum y_n}{1-\hat{y}}\right]$$

$$= -\left[\frac{\frac{1}{N}\sum y_n - \frac{\hat{y}}{N}\sum y_n - \hat{y} + \frac{\hat{y}}{N}\sum y_n}{\hat{y}(1-\hat{y})}\right]$$

$$= -\left[\frac{\frac{1}{N}\sum y_n - \hat{y}}{\hat{y}(1-\hat{y})}\right] = \frac{\hat{y}-\nu}{\hat{y}(1-\hat{y})} = 0$$

$$\Rightarrow \hat{y} - \nu = 0$$

$$\hat{y} = \nu \ \#$$ Yes

6. (A)

Stochastic Gradient Descent

good

A

6. In the perceptron learning algorithm, we find one example $(\mathbf{x}_{n(t)}, y_{n(t)})$ that the current weight vector $\mathbf{w}_t$ mis-classifies, and then update $\mathbf{w}_t$ by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}.$$

The algorithm can be viewed as optimizing some $E_{\text{in}}(\mathbf{w})$ that is composed of one of the following point-wise error functions with stochastic gradient descent (neglecting any non-differentiable points of the error function). What is the error function? Choose the correct answer; explain your answer.

[a] $\text{err}(\mathbf{w}, \mathbf{x}, y) = \max(0, -y\mathbf{w}^T\mathbf{x})$

[b] $\text{err}(\mathbf{w}, \mathbf{x}, y) = \cancel{-\max(0, -y\mathbf{w}^T\mathbf{x})}$

[c] $\text{err}(\mathbf{w}, \mathbf{x}, y) = \max(y\mathbf{w}^T\mathbf{x}, -y\mathbf{w}^T\mathbf{x})$

[d] $\text{err}(\mathbf{w}, \mathbf{x}, y) = -\max(y\mathbf{w}^T\mathbf{x}, -y\mathbf{w}^T\mathbf{x})$

[e] none of the other choices

if correct

err = 0

that is yw^Tx 同号

→ 正

(as $\max(0. -yw^Tx)$

if error (y, w^Tx)異號

$-(yw^Tx)$

7.(A)

$$\sum_{k=0} \ln [y=k] \frac{e^{w^T x}}{\sum_{j=0} e^{w^T x}}$$

取完 gradient 要加負號

→ 正確分類為 k 時對 err 沒有貢獻 ($\ln 1 = 0$)

⇒ 展開 $hk = \sum_{k=0} [y=k] \ln e^{w^T x} - [y=k] \ln \sum_{j=0} e^{w^T x}$

$= \sum_{k=0} [y=k] w^T x - \sum_{k=0} [y=k] \ln \sum e^{w^T x}$

$\sum (y=k) w^T x = (y=0) w^T k + (y=1) w^T k + \dots (y=k) w^T k + \dots \Rightarrow (y=k) w^T x$

(只有 y=k 有值，that is 樣本正確類別，其它 =0)

同理 $\sum (y=k) \ln \sum_{j=0} e^{w^T x} = \ln \sum_{j=0} e^{w^T x}$

$h(k) = [(y=k) w^T x - \ln \sum_{j=0} e^{w^T x}]$

$$\frac{\partial h(k)}{\partial w_t} = [x - \frac{1}{\sum e^{w^T x}} e^{w^T x} \cdot x]$$

t=c

$= -x(1 - \frac{e^{w^T x}}{\sum e^{w^T x}})$

t≠c

$= -x(1 - h_y)$

$= (0 - \frac{1}{\sum e^{w^T x}} e^{w^T x} \cdot x) = -x(-\frac{e^{w^T x}}{\sum e^{w^T x}})$

$= x(h_y)$

$$\nabla \frac{\partial h(k)}{\partial w_t} = (h_y - 1(y=k)) x$$

$$w \Rightarrow w + (1(y=k) - h_y) x$$

這邊應該要選 y=k

→ $(1 - h_y) x$ → A

8. (E)

E

Nonlinear Transformation

8. Given the following training data set:

$$\mathbf{x}_1 = (0,1), y_1 = -1 \quad \mathbf{x}_2 = (0,-1), y_2 = -1 \quad \mathbf{x}_3 = (-1,0), y_3 = +1 \quad \mathbf{x}_4 = (1,0), y_4 = +1$$

Use the quadratic transform $\mathbf{\Phi}_2(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$ and take $\mathrm{sign}(0) = 1$. Which of the following weight vector $\tilde{\mathbf{w}}$ represents a linear classifier in the $\mathcal{Z}$-space that can separate all the transformed examples perfectly? Choose the correct answer; explain your answer.

[a] $(0, -1, 0, 0, 0, 0)$

[b] $(0, 0, -1, 0, 0, 0)$

[c] $(0, 0, 0, -1, 0, 0)$

[d] $(0, 0, 0, 0, -1, 0)$

[e] $(0, 0, 0, 0, 0, -1)$

for $x_1 = [1, 0, 1, 0, 0, 1]$

$x_2 = [1, 0, -1, 0, 0, 1]$

$x_3 = [1, -1, 0, 1, 0, 0]$

$x_4 = [1, 1, 0, 1, 0, 0]$

$W_a \cdot X_1 = 1$ wrong

$W_b \cdot X_2 = 1$ wrong

$W_c \cdot X = 1$ wrong

W_d correct

$W_e =$

$(\Gamma x)^T$

$W = \ldots \hat{x}^T y$ $= x^T \Gamma^T$

4 of 7

9.(B(

$W = X P \cdot Y$

$\hat{w} = (Px)^{-1} Y$

$(Px)^T \hat{w} = Y$

$\rightarrow$

$x^T W_{lin} = Y$

$x^T W_{lin} = (Px)^T \hat{w}$

$x^T W_{lin} = x^T P^T \hat{w}$

$W_{lin} = P^T \hat{w}$

10.(C)

C

10. After "visualizing" the data and noticing that all $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$ are distinct, Dr. Trans magically decides the following transform

$E_{in}(\tilde{w}) =$

$$\mathbf{\Phi}(\mathbf{x}) = ([\![\mathbf{x} = \mathbf{x}_1]\!], [\![\mathbf{x} = \mathbf{x}_2]\!], \ldots, [\![\mathbf{x} = \mathbf{x}_N]\!]).$$

if $x_n = x$ then $\rightarrow 1$

otherwise $= 0$.

That is, $\mathbf{\Phi}(\mathbf{x})$ is a N-dimentional vector whose n-th component is 1 if and only if $\mathbf{x} = \mathbf{x}_n$. If we run linear regression after applying this transform, what is the optimal $\tilde{\mathbf{w}}$? Choose the correct answer; explain your answer.

if $x = [14, 15, 10]$ $\rightarrow$ that is the data after transform

$\Phi(15) = [0, 1, 0]$

$\rightarrow [0, 0, 0, 0, 0, \ldots 1, 0, 0, 0, 0]$.

[a] **1**, the vector of all 1s.

[b] **0**, the vector of all 0s.

[c] $\mathbf{y}$

[d] $-\mathbf{y}$

[e] none of the other choices

$\tilde{w} = (\tilde{x}^T \tilde{x})^{-1} \tilde{x}^T y$

(Note: Be sure to also check what $E_{in}(\tilde{\mathbf{w}})$ is!)

$\tilde{w} = (x^T x)^{-1} x^T y \rightarrow \tilde{w} = 1 \cdot y = y$.

11. Assume that we couple linear regression with one-versus-all decomposition for multi-class classifi-

11.(C)

11. Assume that we coulple linear regression with one-versus-all decomposition for multi-class classification, and get K weight vectors $\mathbf{w}^*_{[k]}$. Assume that the squared error $E^{\text{sqr}}_{\text{in}}(\mathbf{w}^*_{[k]})$ for the k-th binary classification problem is e_k. What is the tightest upper bound of $E^{0/1}_{\text{in}}(g)$, where g is the multi-class classifier formed by the one-versus-all decomposition? Choose the correct answer; explain your answer.

[a] $2\sum_{k=1}^{K} e_k$

[b] $\sum_{k=1}^{K} e_k$

[c] $\frac{1}{2}\sum_{k=1}^{K} e_k$

[d] $\frac{1}{K}\sum_{k=1}^{K} e_k$

[e] $\frac{1}{2K}\sum_{k=1}^{K} e_k$

多類別分類 { linear, one-versus-all →

$E^{sqr}_{in}[w^*_k)$

$= \frac{1}{K}\sum_{n=1}^{K}[w^{*T}_k x_n - y_n)^2 = e_k$

OVA最後的 prediction 來自於

最少有幾個 classifier 犯錯

每個 $w_{(k)}^T x$

OVA $g(x) = \text{argmax}_k$ → 選分類最好的 機率提供最大的

$\text{sign}[w_k^T x_n \times y_n]$

TA hour !!

$N \cdot E^{01}(g) \le N \sum_{1}^{K} E^{01}_{in}(w_k)/2$

$\le N \sum_{1}^{K} E^{sqr}_{in}(w_k)/2$

12. 13.14.15.16.

```
import numpy as np
import random

def getdata(a):
        text =[]
        if a ==1:
            path = 'hw3_train.dat.txt'
        elif a ==2:
            path = 'hw3_test.dat.txt'
        with open(path) as f:
           for line in f:
               text.append([float(i) for i in line.split()])
        mm=np.asarray(text)
        X=mm[:,:-1]
        Y=mm[:,-1]

        return X,Y


        return X,Y
def functionQ(X , q):
    function = []
    func_temp =[1]
    for i in X:
        for j in range(1,q+1):
            for x in i[0:]:
                func_temp.append(x**j)
        function.append(func_temp)
        func_temp = [1]
    function_array = np.array(function)
    return function_array

def functionFullQ(X , q):
    function = []
    func_temp =[1]
    row = X.shape[1]
    a = 1
    b = 0
    for i in X:
        for j in range(1,q+1):
            for x in i[0:]:
                func_temp.append(x**j)
                for _ in range(a , row):
                    func_temp.append(x*X[b][_])
                a+=1
        function.append(func_temp)
        func_temp = [1]
        a = 1
        b+=1
    function_array = np.array(function)
    return function_array
```

```
def functionLower(X, q):
    function = []
    func_temp =[1]
    for i in X:
        for x in i[0:q]:
            func_temp.append(x)
        function.append(func_temp)
        func_temp = [1]
    function_array = np.array(function)
    return function_array
def functionRandom(X, n):
    random.seed(n)
    function = []
    func_temp =[1]
    list1 = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
    list2 = [5,6,7,8,9]
    random.shuffle(list1)
    list3 = np.delete(list1,list2)
    for i in range(X.shape[0]):
        newX = np.delete(X[i],list3)
        for _ in range(newX.shape[0]):
            func_temp.append(newX[_])
        function.append(func_temp)
        func_temp = [1]

    function_array = np.array(function)
    return function_array


def linear_regression(x , y):
    x_t =np.transpose(x)
    t = x_t.dot(x)
    t_inv = np.linalg.inv(t)
    w_lin = (t_inv.dot(x_t)).dot(y)
    return w_lin

def E_in_minus_E_out_bin(w, x, y, x_test, y_test,n):

    # E_in_bin
    E_in_bin = 0
    N = y.size
    for xn, yn in zip(x, y):
        E_in_bin += (np.sign(xn.dot(w)) != yn)/N
    if n :
        print("E_in_bin:", E_in_bin)
    # E_out_bin
    E_out_bin = 0
    N_test = y_test.size
    for xn, yn in zip(x_test, y_test):
        E_out_bin += (np.sign(xn.dot(w)) != yn)/N_test
    if n :
        print("E_out_bin:", E_out_bin)
        print ("|E_in_bin - E_out_bin|:", abs(E_in_bin - E_out_bin))
    if n == 0:
        return abs(E_in_bin - E_out_bin)
```

```python
x_train,y_train= getdata(1)
x_test,y_test = getdata(2)

print('q12')
x_train_trans_2 = functionQ(x_train , 2)
x_test_trans_2 =functionQ(x_test, 2)
wlin_train_2 = linear_regression(x_train_trans_2, y_train)
E_in_minus_E_out_bin(wlin_train_2, x_train_trans_2, y_train, x_test_trans_2, y_test,1)

print('q13')
x_train_trans_8 = functionQ(x_train , 8)
x_test_trans_8 =functionQ(x_test, 8)
wlin_train_8 = linear_regression(x_train_trans_8, y_train)
E_in_minus_E_out_bin(wlin_train_8, x_train_trans_8, y_train, x_test_trans_8, y_test,1)

print('q14')
x_train_fulltrans_2 = functionFullQ(x_train, 2)
x_test_fulltrans_2 = functionFullQ(x_test, 2)
wlin_fulltrain_2 = linear_regression(x_train_fulltrans_2, y_train)
E_in_minus_E_out_bin(wlin_fulltrain_2, x_train_fulltrans_2, y_train, x_test_fulltrans_2, y_test,1)

print('q15')
x_train_compose =[]
for i in range(1,11):
    x_train_c = functionLower(x_train, i)
    x_test_c = functionLower(x_test, i)
    w_c = linear_regression(x_train_c, y_train)
    temp = E_in_minus_E_out_bin(w_c, x_train_c, y_train, x_test_c, y_test,0)
    x_train_compose.append(temp)
print("The minimum of i is ",x_train_compose.index(min(x_train_compose))+1)
print("|E_in_bin - E_out_bin|:",min(x_train_compose))

print('q16')
temprandom=0
for _ in range(200):
    n = random.random()
    x_train_random =functionRandom(x_train, n)
    x_test_random = functionRandom(x_test, n)
    w_random = linear_regression(x_train_random, y_train)
    temprandom += E_in_minus_E_out_bin(w_random, x_train_random, y_train, x_test_random, y_test, 0)
print("the average |E_in_bin - E_out_bin|over 200 experiments :" , temprandom/200)
```

```
In [145]: runfile('C:/Users/paddy/Desktop/NTU_course/2021_fall/HTML/homework3
wdir='C:/Users/paddy/Desktop/NTU_course/2021_fall/HTML/homework3')
q12
E_in_bin: 0.18100000000000013
E_out_bin: 0.507333333333344
|E_in_bin - E_out_bin|: 0.3263333333333438
q13
E_in_bin: 0.05200000000000004
E_out_bin: 0.509666666666677
|E_in_bin - E_out_bin|: 0.457666666666677
q14
E_in_bin: 0.16800000000000012
E_out_bin: 0.5066666666666774
|E_in_bin - E_out_bin|: 0.3386666666666772
q15
The minimum of i is  3
|E_in_bin - E_out_bin|: 0.13233333333334057
q16
the average |E_in_bin - E_out_bin|over 200 experiments : 0.20759666666667237
```