Machine Learning HW3

B06502152, 許書銓

---

1. The answer should be (b).

$$\mathbb{E}_D[E_{in}(w_{in})] = \sigma^2(1 - \frac{d+1}{N}) \tag{1.0}$$

   Now, we would like to know how big should N be so that $\mathbb{E}_D[E_{in}(w_{in})] \geq 0.006$.

$$\frac{\mathbb{E}_D[E_{in}(w_{in})]}{\sigma^2} \leq 1 - \frac{d+1}{N} \tag{1.1}$$

$$\frac{1}{N} \leq \frac{1}{d+1}(1 - \frac{\mathbb{E}_D[E_{in}(w_{in})]}{\sigma^2}) \tag{1.2}$$

$$N \geq \frac{1}{\frac{1}{d+1}(1 - \frac{\mathbb{E}_D[E_{in}(w_{in})]}{\sigma^2})} \tag{1.3}$$

   Now, we plug in all numbers, which $\sigma = 0.1$, $d = 11$ and $\mathbb{E}_D[E_{in}(w_{in})] = 0.006$, N will

$$\begin{aligned} N &\geq \frac{1}{\frac{1}{12}(1 - \frac{0.006}{0.01^2})} \\ &= 30 \end{aligned} \tag{1.4}$$

2. The answer should be (a).

   (a.) To prove the choice we have to know the following propositions,

Proposition 1: For $v \in R^n$, $A^T Av = 0$ if and only if $Av = 0$

proof : if $A^T Av = 0$, $Av = 0$

$$v^T(A^T A)v = (Av)^T(Av) = ||Av||^2 \tag{2.0}$$

if $A^T Av = 0$

$$v^T(A^T A)v = v^T * 0 = 0 = ||Av||^2 \tag{2.1}$$

hence, $Av = 0$.

proof: if $Av = 0$, $A^T Av = 0$

$$A^T * 0 = 0 \tag{2.2}$$

Proposition 2: The normal equation exist at least one solutions.

For a linear system, that $My = c$ has a solution if and only if when $M^T v = 0$, $c^T v = 0$.

Here, let $M^T = A^T A$ and $c = A^T b$. Suppose $M^T v = 0 = A^T Av$. From proposition 1, we know that $Av = 0$. Then $c^T v = (A^T b)^T v = b^T Av = 0$. We prove that there has a solution!

(b.)(c.) Since if we rearrange the equation, we could have,

$$\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y}$$

If $(X^T X)$ is invertible, we would have a unique solution for $\nabla E_{in} = 0$. However, we can't guarantee that $E_{in} = 0$, since it can only represent that E_{in} is at its minimum.

(d.) we cannot guarantee that there exist only one unique solution, since there may exist infinite solutions for the equation for some particular X and y.

3. The answer should be (c)

(a). If X is multiplying by 2, then we could rewrite H matrix as,

$$
\begin{aligned}
H &= cX(\ (cX)^T(cX)\ )^{-1}(cX)^T \\
&= cX(cX^TcX)^{-1}cX^T && ((cX)^T = cX^T) \\
&= c^2X(c^2X^TX)^{-1}X^T \\
&= c^2X(\frac{1}{c^2})(X^TX)^{-1}X^T && ((cX)^{-1} = c^{-1}X^{-1}) \\
&= X(X^TX)^{-1}X^T
\end{aligned}
$$

Hence, we can see that H will not changed.

(b). Since as we know, H works as a projection matrix. Hence, we can refer this question as will the operation would modified the span(X). If multiplying each of the i-th column of X by i, then span(X) will not change since every column just simply multipled by a scalar the normalized column vector is not changed.

(c). In this choice, the span(X) may changed owning to multyplying each of the n-th row of X by $1/n$. We can give a example, considering two vector $\mathbf{x_1} = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$, and $\mathbf{x_2} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$. After operations, we have $\mathbf{x'_1} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$, and $\mathbf{x'_2} = \begin{bmatrix} 2 \\ 3/2 \\ 1/3 \end{bmatrix}$. We can see that $\mathbf{x1} \times \mathbf{x2} \neq \mathbf{x1}' \times \mathbf{x2}'$. Hence, span(X) would change.

(d). Since as we know, H works as a projection matrix. Hence, we can refer this question as will the operation would modified the span(X). If adding three randomly-chosen i,j,k to column 1 of X, then span(X) will not change since .

4. The answer should be (e).

- $Pr(|\nu - \theta| > \epsilon) \leq 2exp(-2\epsilon^2 N)$ for all $N \in \mathbb{N}$ and $\epsilon > 0$.

  It is a correct Hoeffding's equation when we look a target function.

- ν maximizes likelihood(θ) over all $\theta \in [0, 1]$.

  The function of the probability is

$$
f(x) = \Pi_{i=1}^{n} p^{x_i}(1-p)^{1-x_i} \tag{4.0}
$$

Here, we would like to find the max of the function. Moreover, we know that log operation is a monotonic function, which means that if we take a function a log operation, a maximun place will remain the same. Hence, we change our function as

$$
\begin{aligned}
g(x) &= \Sigma_{i=1}^{n} \ln(p^{x_i}(1-p)^{1-x_i}) \\
&= \Sigma_{i=1}^{n} \ln(p^{x_i}) + \Sigma_{i=1}^{n} \ln(1-p)^{1-x_i} \\
&= \Sigma_{i=1}^{n} x_i \ln(p) + (n - \Sigma_{i=1}^{n} x_i) \ln(1-p)
\end{aligned} \tag{4.1}
$$

And we can find the maximun of g(x) if we take derivative of g(x)

$$g'(x) = \frac{\Sigma_{i=1}^{n} x_i}{p} + \frac{(n - \Sigma_{i=1}^{n} x_i)}{1-p} \tag{4.2}$$

When $\frac{\Sigma_{i=1}^{n} x_i}{p} = \frac{(n-\Sigma_{i=1}^{n} x_i)}{1-p}$, we can find that p = $\frac{1}{N}(\Sigma_{i=1}^{N} x_i)$, which is ν.

- ν minimizes $E_{in}(\hat{y}) = \frac{1}{N}\Sigma_{n=1}^{N}(\hat{y} - y_n)^2$ over all $\hat{y} \in \mathbb{R}$.

$$E'_{in}(\hat{y}) = \frac{1}{N}\,\Sigma_{n=1}^{N} 2(\hat{y} - y_n)$$

We would like to $E_{in} = 0$,

$$\begin{aligned} n \times \hat{y} &= \Sigma_{n=1}^{N} y_n \\ \hat{y} &= \frac{1}{N}\Sigma_{n=1}^{N} y_n = \nu \end{aligned} \tag{4.3}$$

- $2 \cdot \nu$ is the negative gradient direction $-\nabla E_{in}(\hat{y})$ at $\hat{y} = 0$.

$$\begin{aligned} -\nabla E_{in}(\hat{y}) &= \frac{1}{N}\,\Sigma_{n=1}^{N} 2(\hat{y} - y_i) \\ &= -2\Sigma_{n=1}^{N} y_i \\ &= -2\nu \end{aligned} \tag{$\hat{y} = 0$}$$

Hence, all the scenrios are correct.

5. The answer should be (a).

Since $y_1, y_2, \ldots, y_n$ are from uniform distribution, we know the probability density function of

$$f(x) = \begin{cases} \dfrac{1}{b-a}, & a \leq x \leq b \\ 0, & x \leq a \; or \; x \geq b \end{cases}$$

Here, we would like use $\hat{\theta}$ to estimate the likelihood. With N i.i.d data, we can estimate the likelihood as $(\frac{1}{\hat{\theta}})^N$.

6. The answer should be (b).

From the lecture note, we know that PLA will change when $y_n \neq w_t^T x_n$, the situation can rewrite as if

$$err(w, x, y) = \begin{cases} 1, & y_n \neq w_t^T x_n \quad (y_n w_t^T x_n < 0) \\ 0, & y_n = w_t^T x_n \quad (y_n w_t^T x_n > 0) \end{cases}$$

Furthermore, we can transorm the $err(w, x, y) = max(0, -y_n w_t^T x_n)$.

7. The answer should be (a).

We realize that $err_{exp}(\mathbf{w}, \mathbf{x}, y) = exp(-y\mathbf{w^T x})$. Now, we can calculate its gradient by

$$\nabla err_{exp}(\mathbf{w}, \mathbf{x}, y) = -y_n \mathbf{x_n} exp(-y\mathbf{w^T x}) \tag{7.0}$$

Hence,

$$-\nabla err_{exp}(\mathbf{w}, \mathbf{x}, y) = +y_n \mathbf{x_n} exp(-y\mathbf{w^T x}) \tag{7.1}$$

8. The answer should be (b).

Here, we would like to know the optimal direction v to minimize E(w). From the idea of Taylor Expansion, we can consider the equation of Taylor Expansion is based on the use a really close point to compute the data we want. As we can see from the question

$$\begin{aligned} E(\mathbf{w}) &= E(\mathbf{u}) + \mathbf{b}_E(\mathbf{u})^T(\mathbf{w} - \mathbf{u}) + \frac{1}{2}(\mathbf{w} - \mathbf{u})^T A_E(\mathbf{u})(\mathbf{w} - \mathbf{u}) && (8.0) \\ &= E(\mathbf{u}) + \mathbf{b}_E(\mathbf{u})^T(\mathbf{v}) + \frac{1}{2}(\mathbf{v})^T A_E(\mathbf{u})(\mathbf{v}) && (8.1) \end{aligned}$$

The fist term of the right hand side of equation 8.0 is a fixed value, thus we aim to minimize the latter terms, which equals the terms with $\mathbf{v}$ in equation 8.1 . Hence, we would like to find the minimun of it. Let's say the later term as E_{later}, by taking derivative

$$\begin{aligned} \nabla E_{later} &= \mathbf{b}_E(\mathbf{u})^T + \frac{1}{2}A_E(\mathbf{u})(\mathbf{v}) + \frac{1}{2}(\mathbf{v})^T A_E(\mathbf{u}) && (8.2) \\ &= \mathbf{b}_E(\mathbf{u})^T + A_E(\mathbf{u})(\mathbf{v}) = 0 && (8.3) \end{aligned}$$

Thus, we get that

$$\mathbf{v} = -(A_E(\mathbf{u}))^{-1}\mathbf{b}_E(\mathbf{u}) \tag{8.4}$$

9. The answer should be (b).

$$E_{in} = \frac{1}{N}||\mathbf{x}\mathbf{w} - \mathbf{y}||^2 = \frac{1}{N}(\mathbf{w}^T\mathbf{x}^T\mathbf{x}\mathbf{w} - 2\mathbf{w}^T\mathbf{x}^T\mathbf{y} + \mathbf{y}^T\mathbf{y}) \tag{9.0}$$

We calculate the first gradient

$$\nabla E_{in} = \frac{2}{N}(\mathbf{x}^T\mathbf{x}\mathbf{w} - \mathbf{x}^T\mathbf{y}) \tag{9.1}$$

We then calculate the Hessina matrix,

$$\nabla^2 E_{in} = \frac{2}{N}(\mathbf{x}^T\mathbf{x}) \tag{9.3}$$

10. The answer should be (b).

To maximize the likelihood, we have to minimize the error function. From the error function,

$$\begin{aligned} err(W, \mathbf{x}, y) = -\ln h_y(\mathbf{x}) &= -\Sigma_{k=1}^{K}[|y = k|]\ln(h_k(\mathbf{x})) \\ &= \ln(\Sigma_{i=1}^{K}\exp(\mathbf{w_i^T}\mathbf{x})) - [|y = k|]\ln(\exp(\mathbf{w_y}^T\mathbf{x}) \\ &= \ln(\Sigma_{i=1}^{K}\exp(\mathbf{w_i^T}\mathbf{x})) - [|y = k|]\mathbf{w_y}^T\mathbf{x} \end{aligned} \tag{10.0}$$

Next step, we try to find the conditon when $\frac{\partial err(W,\mathbf{x},y)}{\partial \mathbf{w_i k}} = 0$.

$$\begin{aligned} \frac{\partial err(W, \mathbf{x}, y)}{\partial \mathbf{w_{ik}}} &= \frac{\partial[\ \ln(\Sigma_{i=1}^{K}\exp(\mathbf{w_i^T}\mathbf{x_i})) - [|y = k|]\mathbf{w_y}^T\mathbf{x_i}\ ]}{\partial \mathbf{w_{ik}}} \\ &= \frac{\exp(\mathbf{w_k^T}\mathbf{x_i})\mathbf{x_i}}{\ln(\Sigma_{i=1}^{K}\exp(\mathbf{w_i^T}\mathbf{x_i}))} - [|y = k|]\mathbf{x_i} \\ &= (\ h_k(\mathbf{x}) - [|y = k|]\ )\mathbf{x_i} \end{aligned}$$

11. The answer should be (e)

From the definition of h_y , we can rewrite it as,

$$
\begin{aligned}
h_1 &= \frac{e^{\mathbf{w}_1^T \mathbf{x}}}{e^{\mathbf{w}_1^T \mathbf{x}} + e^{\mathbf{w}_2^T \mathbf{x}}} \\
&= \frac{1}{1 + e^{-(\mathbf{w}_1^T - \mathbf{w}_2^T)\mathbf{x}}} && (11.0) \\
h_2 &= \frac{e^{\mathbf{w}_2^T \mathbf{x}}}{e^{\mathbf{w}_1^T \mathbf{x}} + e^{\mathbf{w}_2^T \mathbf{x}}} \\
&= \frac{1}{1 + e^{-(\mathbf{w}_2^T - \mathbf{w}_1^T)\mathbf{x}}} && (11.1)
\end{aligned}
$$

Here, we want to find the optimal solution of the logistic regression, hence we try to maximize the likelihood , which $\propto \Pi_{n=1}^{N} h_{y_n}(y_n x_n)$.

Moreover, from the relationship to maximize the likelihood equal to minimize "Cross Entropy Error".

Since k will be satisfied either when k = 1 or k = 2. Here, we first try to pick as k = 1. And $y' = 2y - 3$

$$
\begin{aligned}
\min \quad -\frac{1}{N}\Sigma_{n=1}^{N} \ln(h_1(\mathbf{x}, y_1)) &= \frac{1}{1 + e^{-y_1(\mathbf{w}_1^* - \mathbf{w}_2^*)\mathbf{x}}} \\
&= \frac{1}{1 + e^{-(\mathbf{w}_2^* - \mathbf{w}_1^*)\mathbf{x}}} && (11.2)
\end{aligned}
$$

Hence, the optimal solution is $(\mathbf{w_2^*} - \mathbf{w_1^*})$ from the p12. of lecture 10.

12. The answer should be (e).

    if we try to compute the output by the the following code

```python
def compute(x ,c , a, b, aa, ab, bb):
    return c*1 + a * x[0] + b * x[1] + aa * x[0] * x[0] + ab * x[0] * x[1] + bb * x[1] * x[1]


c , a, b, aa, ab, bb = input('Enter input coefficients: ').split(',')
c = int(c)
a = int(a)
b = int(b)
aa = int(aa)
ab = int(ab)
bb = int(bb)

x1 = (0, 1)
x2 = (1, -1)
x3 = (-1, 0)
```

```python
x4 = (-1, 2)
x5 = (2, 0)
x6 = (1, -1.5)
x7 = (0, 2)

print(np.sign(compute(x1, c , a, b, aa, ab, bb)))
print(np.sign(compute(x2, c , a, b, aa, ab, bb)))
print(np.sign(compute(x3, c , a, b, aa, ab, bb)))
print(np.sign(compute(x4, c , a, b, aa, ab, bb)))
print(np.sign(compute(x5, c , a, b, aa, ab, bb)))
print(np.sign(compute(x6, c , a, b, aa, ab, bb)))
print(np.sign(compute(x7, c , a, b, aa, ab, bb)))
```

(a). [-1, -1, -1, 1, -1, 1, 1]

(b). [-1, 1, -1, 1, -1, 1, 1]

(c). [1, 1, 1, 1, 1, 1]

(d). [-1, -1, -1, -1, -1, -1]

(e). [-1, -1, -1, 1, 1, 1, 1]

13. The answer should be (b)

As we know for the definition of VC dimension, some set of N, which $N \leq d_{vc}$ should be shattered. That is, the grow function of the hypothesis when $N \leq d_{vc}$ should fulfill that,

$$grow\ function = 2^{d_{vc}}, where\ N \leq d_{vc} \tag{13.0}$$

For the hypothesis here, we can somewhat view it as a decision stump of d dimension, since we can view $c_0 * 1$ from $(1, x_k)$ as threshold and $c_1 * x_k$ is larger then $c_0 * 1$ or not. Hence, from the outcome of problem 2 of hw2. We realize that the grow function of this kind of decision stump is

$$grow\ function = 2(N-1)d \tag{13.1}$$

From the equation 13.0 and 13,1, we could write down,

$$
\begin{aligned}
2(N-1)d &= 2^{d_{vc}} \leq 2Nd\,, \quad (N \leq d_v c) \\
2Nd &\geq 2^{d_{vc}} \\
\frac{d_{vc}}{2} + \frac{d}{2} &\geq \log_2 d_{vc} + \frac{d}{2} \geq d_{vc} - 1 \\
d_{vc} - \frac{d_{vc}}{2} &\leq 1 + \frac{d}{2} \\
\frac{d_{vc}}{2} &\leq 1 + \frac{d}{2} \leq 1 + \log_2 d \\
d_{vc} &\leq 2\left(1 + \log_2 d\right) \qquad (13.2)
\end{aligned}
$$

14. The answer should be (d).

```python
##for question 14
import numpy as np

#main
data_in = np.genfromtxt("hw3_train.dat.txt")

X = data_in[:, : -1]
y = data_in[:, -1]
X = np.c_[(1 * np.ones(X.shape[0]), X)]

X_p = np.linalg.pinv(X)
w_lin = X_p.dot(y)
E_in_sqr = 0
tmp_s = np.inner(X, w_lin)-y
E_in_sqr = np.linalg.norm(tmp_s)**2/X.shape[0]
print("E_in = ", E_in_sqr)
```

15. The answer should be (c).

```python
##for question 15
import numpy as np
import random
import py_compile
import matplotlib.pyplot as plt

#main
data = np.genfromtxt("hw3_train.dat.txt")

result = []
X = data[:, : -1]
y = data[:, -1]
y = y.reshape([X.shape[0], 1])
X = np.c_[(1 * np.ones(X.shape[0]), X)]
```

```python
X_p = np.linalg.pinv(X)
w_lin = X_p.dot(y)

E_in_sqr = 0
for i in range(X.shape[0]):
    x_i = np.array(X[i, :])
    x_i = x_i.reshape((1, X.shape[1]))
    E_in_sqr += (np.dot(x_i,w_lin) - y[i]) * (np.dot(x_i,w_lin) - y[i])

E_in_sqr /= X.shape[0]
#print(E_in_sqr)

for i in range(1000):
    w = np.zeros((X.shape[1], 1), np.float)
    t = 0 #count the number of iteration
    #print(i)
    while True:
        t += 1
        k = random.randint(0, X.shape[0]-1)
        x_k = np.array(X[k, :])
        x_k = x_k.reshape((1, X.shape[1]))

        w_gradient=np.zeros(shape=(X.shape[1], 1))
        prediction=np.dot(x_k, w)
        w_gradient = (-2)*(y[k]-(prediction)) * x_k.T

        w = w -  0.001 * w_gradient

        w_s = np.squeeze(w)
        y = np.squeeze(y)
        temp = np.inner(X, w_s) - y
        E_in = np.linalg.norm(temp)**2
        E_in /= X.shape[0]

        if E_in <= 1.01 * E_in_sqr:
            #print(t)
            break

    result.append(t)

plt.hist(result)
plt.show()
print("The average number of iteration is : ", np.mean(result))
```

16. The answer should be (c).

```python
##for question 16
import numpy as np
import random
```

```
import py_compile
import math
import matplotlib.pyplot as plt

#main
data = np.genfromtxt("hw3_train.dat.txt")

result = []
X = data[:, : -1]
y = data[:, -1]
y = y.reshape([X.shape[0], 1])
X = np.c_[(1 * np.ones(X.shape[0]), X)]
X_p = np.linalg.pinv(X)
w_lin = X_p.dot(y)

E_in_sqr = 0
for i in range(X.shape[0]):
    x_i = np.array(X[i, :])
    x_i = x_i.reshape((1, X.shape[1]))
    E_in_sqr += (np.dot(x_i,w_lin) - y[i]) * (np.dot(x_i,w_lin) - y[i])

E_in_sqr /= X.shape[0]
#print(E_in_sqr)

for i in range(1000):
    w = np.zeros((X.shape[1], 1), np.float)
    t = 0 #count the number of iteration

    for j in range(500):
        t += 1
        k = random.randint(0, X.shape[0]-1)
        x_k = np.array(X[k, :])
        x_k = x_k.reshape((1, X.shape[1]))

        s = -y[k] * np.dot(x_k, w)
        w = w +  0.001 / (1 + math.exp(-s))  * y[k] * x_k.T

    E_in = 0
    s = np.dot(X, w)
    for j in range(X.shape[0]):
        E_in += math.log((1 + math.exp(-y[j]*s[j])),math.e)
    E_in /= X.shape[0]
    result.append(E_in)

plt.hist(result)
plt.show()
print("The average E_in_500 is : ", np.mean(result))
```

17. The answer should be (b).

```python
##for question 17
import numpy as np
import random
import py_compile
import math
import matplotlib.pyplot as plt

#main
data = np.genfromtxt("hw3_train.dat.txt")

result = []
X = data[:, : -1]
y = data[:, -1]
y = y.reshape([X.shape[0], 1])
X = np.c_[(1 * np.ones(X.shape[0]), X)]
X_p = np.linalg.pinv(X)
w_lin = X_p.dot(y)

E_in_sqr = 0
for i in range(X.shape[0]):
    x_i = np.array(X[i, :])
    x_i = x_i.reshape((1, X.shape[1]))
    E_in_sqr += (np.dot(x_i,w_lin) - y[i]) * (np.dot(x_i,w_lin) - y[i])

E_in_sqr /= X.shape[0]
#print(E_in_sqr)

for i in range(1000):
    w = w_lin
    t = 0 #count the number of iteration

    for j in range(500):
        t += 1
        k = random.randint(0, X.shape[0]-1)
        x_k = np.array(X[k, :])
        x_k = x_k.reshape((1, X.shape[1]))

        s = -y[k] * np.dot(x_k, w)
        w = w +  0.001 / (1 + math.exp(-s))  * y[k] * x_k.T

    E_in = 0
    s = np.dot(X, w)

    for j in range(X.shape[0]):
        E_in += math.log((1 + math.exp(-y[j]*s[j])),math.e)
    E_in /= X.shape[0]
    #print(E_in)
    result.append(E_in)

plt.hist(result)
plt.show()
```

```python
print("The average E_in_500 is : ", np.mean(result))
```

18. The answer should be (a).

```python
##for question 18
import numpy as np

#main
data_in = np.genfromtxt("hw3_train.dat.txt")

X = data_in[:, : -1]
y = data_in[:, -1]
X = np.c_[(1 * np.ones(X.shape[0]), X)]

X_p = np.linalg.pinv(X)
w_lin = X_p.dot(y)
s = np.dot(X, w_lin)

E_in = 0
for i in range (X.shape[0]):
    if(np.sign(s[i]) != y[i]):
        E_in += 1
E_in /= X.shape[0]

data_out = np.genfromtxt("hw3_test.dat.txt")
X_out = data_out[:, : -1]
y_out = data_out[:, -1]
X_out = np.c_[(1 * np.ones(X_out.shape[0]), X_out)]
s_out = np.dot(X_out, w_lin)

E_out = 0
for i in range (X_out.shape[0]):
    if(np.sign(s_out[i]) != y_out[i]):
        E_out += 1
E_out /= X_out.shape[0]
print("|E_in - E_out| = ", abs(E_in - E_out))
```

19. & 20. The answer should be (b), (d).

```python
##for question 19&20
import numpy as np
import math

#main
n = input('Q = ')
n= int(n)

data_in = np.genfromtxt("hw3_train.dat.txt")

```

```python
X = data_in[:, : -1]
d = X.shape[1]
y = data_in[:, -1]
X = np.c_[(1 * np.ones(X.shape[0]), X)]

X_poly = X
for i in range((n - 1)*d):
    exp = math.floor(i / d) + 2
    add = np.power(X_poly[:, i%d+1], exp)
    X_poly = np.insert(X_poly, X_poly.shape[1], values=add, axis=1)

X_p = np.linalg.pinv(X_poly)
w_lin = X_p.dot(y)

s = np.dot(X_poly, w_lin)
E_in = 0
for i in range (X.shape[0]):
    if(np.sign(s[i]) != np.sign(y[i])):
        E_in += 1
E_in /= X.shape[0]
print("E_in = ", E_in)

data_out = np.genfromtxt("hw3_test.dat.txt")
X_out = data_out[:, : -1]
y_out = data_out[:, -1]
X_out = np.c_[(1 * np.ones(X_out.shape[0]), X_out)]

X_out_poly = X_out
for i in range((n - 1)*d):
    exp = math.floor(i / d) + 2
    add = np.power(X_out_poly[:, i%d+1], exp)
    X_out_poly = np.insert(X_out_poly, X_out_poly.shape[1], values=add, axis=1)

s_out = np.dot(X_out_poly, w_lin)
E_out = 0
for i in range (X_out.shape[0]):
    if(np.sign(s_out[i]) != np.sign(y_out[i])):
        E_out += 1
E_out /= X_out.shape[0]
print("E_out = ", E_out)
print("|E_in - E_out| = ", abs(E_in - E_out))
```