Homework # 2

姓名	學號
操之晴	R13922A04

5. The example it gave was wrong. It provided the sequence “1,4,9”, which has 3 terms. According to the question, this sequence should correspond to a polynomial of degree 4 (since N=4, and we need N−1=3 terms) instead of a quadratic polynomial. Additionally, using only 3 equations for 5 unknown coefficients, results in an underdetermined system, meaning there are infinitely many possible solutions, and we can't uniquely predict the next term.
   In conclusion, the answer is correct only when we have at least N+1 terms for an N-degree polynomial. That way, we can set up a system of equations to find the coefficients of the polynomial, which allows us to predict the next term. Otherwise, the next term prediction is unreliable.

6. Since the numbers in A and B won’t be green at the same time, so does C and D, we can see for
   - small even numbers: $P(B \cup D) = P(\text{全}B) + P(\text{全}D) + P(\text{僅}BD)$
   - small odd numbers: $P(A \cup D) = P(\text{全}A) + P(\text{全}D) + P(\text{僅}AD)$
   - big even numbers: $P(B \cup C) = P(\text{全}B) + P(\text{全}C) + P(\text{僅}BC)$
   - big odd numbers: $P(A \cup C) = P(\text{全}A) + P(\text{全}C) + P(\text{僅}AC)$

A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
B	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Thus, the probability on the union set = $P(\text{全}A) + P(\text{全}B) + P(\text{全}C) + P(\text{全}D) + P(\text{僅}AD) + P(\text{僅}BD) + P(\text{僅}AC) + P(\text{僅}BC) = (1+1+1+1+30+30+30+30)/1024 = \mathbf{\frac{31}{256}}$

7. The green 5’s only come from A or D, which the probability is (1+1+30)/1024 $= \mathbf{\frac{1}{32}}$

8. Set $\epsilon = \sqrt{\frac{lnt+lnM-\frac{1}{2}ln\delta}{N_m}}$ then $\exp(-2\epsilon^2 N) = exp(-2lnt - 2lnM + ln\delta) = \frac{\delta}{t^2M^2}$

$\nu = \frac{C_m}{N_m}$. Then we get $P(\mu_m > \frac{C_m}{N_m} + \sqrt{\frac{lnt+lnM-\frac{1}{2}ln\delta}{N_m}}) \leq \frac{\delta}{t^2M^2}$

For all machines, m = 1, 2, 3…M

$$P(\mu_1 > \frac{C_1}{N_1} + \sqrt{\frac{lnt + lnM - \frac{1}{2}ln\delta}{N_1}} \cup \mu_2 > \frac{C_2}{N_2} + \sqrt{\frac{lnt + lnM - \frac{1}{2}ln\delta}{N_2}} \cup \ldots$$

$$\cup \mu_M > \frac{C_M}{N_M} + \sqrt{\frac{lnt + lnM - \frac{1}{2}ln\delta}{N_M}}) \leq \sum_{i=1}^{M} \mathrm{P}(\mu_i > \frac{C_i}{N_i} + \sqrt{\frac{lnt + lnM - \frac{1}{2}ln\delta}{N_i}}) \leq \frac{\delta}{t^2M^2} * M$$

$$= \frac{\delta}{t^2M}$$

For all t=M+1, M+2…

$$\sum_{t=M+1}^{\infty} \sum_{i=1}^{M} \mathrm{P}\left(\mu_i > \frac{C_i}{N_i} + \sqrt{\frac{lnt + lnM - \frac{1}{2}ln\delta}{N_i}}\right) \leq \sum_{t=M+1}^{\infty} \frac{\delta}{t^2M} = \frac{\delta}{M} * \frac{\pi^2}{6}$$

Since M≥ 2, $\frac{\delta}{M}\frac{\pi^2}{6} \leq \delta$. Therefore, for all m= 1, 2, 3…M, all t=M+1, M+2…,

$$P(\mu_m > \frac{C_m}{N_m} + \sqrt{\frac{lnt + lnM - \frac{1}{2}ln\delta}{N_m}}) \leq \delta$$

Which means, with probability at least $1 - \delta$, $\mu_m \leq \frac{C_m}{N_m} + \sqrt{\frac{lnt+lnM-\frac{1}{2}ln\delta}{N_m}}$.

9. Assume k =3, meaning the input has three positions. The possible input combinations are:
   - (-1, -1, -1) : 0 ones
   - (1, -1, -1) : 1 ones
   - (1, 1, -1) : 2 ones
   - (1, 1, 1) : 3 ones

   For a symmetric Boolean function, the value of the function depends only on how many 1s are in the input instead of the positions. So, in k dimensional space, the function can distinguish between k+1 different cases: from 0 ones to k ones (see above 4 cases).

   Therefore, in k dimensional space, a symmetric Boolean function can distinguish up to k+1 distinct patterns, but it cannot shatter more than k+1 points. This means the VC dimension of the set of symmetric Boolean functions is **k+1.**

10. case 1: s = 1、$\theta > 0$、$h(x) = sign(x - \theta)$

$$E_{out}(h_{s=1}, \theta > 0) = \frac{1}{2}P + \frac{\theta}{2}(1 - P) + \frac{1 - \theta}{2}P = P + \frac{1}{2}\theta - \theta P$$

case 2: s = 1、$\theta < 0$、$h(x) = sign(x - \theta)$

$$E_{out}(h_{s=1}, \theta < 0) = \frac{1 + \theta}{2}P + \frac{-\theta}{2}(1 - P) + \frac{1}{2}P = P - \frac{1}{2}\theta + \theta P$$

case 3: s = −1、$\theta > 0$、$h(x) = -sign(x - \theta)$

$$E_{out}(h_{s=-1}, \theta > 0) = \frac{\theta}{2}P + \frac{1}{2}(1 - P) + \frac{1 - \theta}{2}(1 - P) = 1 - P - \frac{1}{2}\theta + \theta P$$

case 4: s = −1、$\theta < 0$、$h(x) = -sign(x - \theta)$

$$E_{out}(h_{s=-1}, \theta < 0) = \frac{-\theta}{2}P + \frac{1}{2}(1 - P) + \frac{1 + \theta}{2}(1 - P) = 1 - P + \frac{1}{2}\theta - \theta P$$

$$E_{out}(h_{s=1}, \theta > 0) + E_{out}(h_{s=1}, \theta < 0) + E_{out}(h_{s=-1}, \theta > 0) + E_{out}(h_{s=-1}, \theta < 0)$$

$$= \frac{1}{2} - s\left(\frac{1}{2} - P - \frac{1}{2}|\theta| + P|\theta|\right) = \frac{1}{2} - s\left(\frac{1}{2} - P\right) + s\left(\frac{1}{2} - P\right)|\theta| = u + v \cdot |\theta|$$

11.

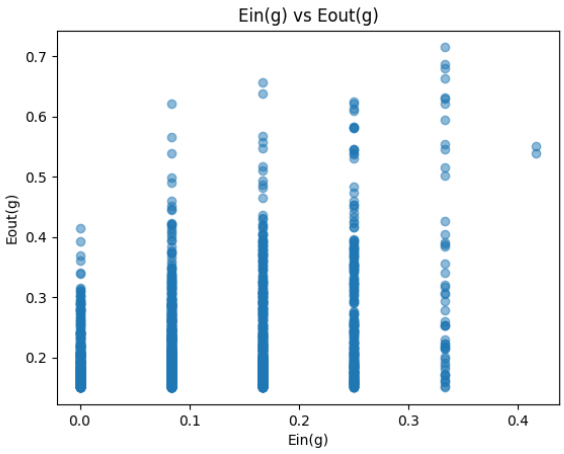

Figure 11a. scatter plot of (Ein(g), Eout(g))

```python
import numpy as np
import matplotlib.pyplot as plt

# generate datas
def generate_data(N,p):
    x = np.sort(np.random.uniform(-1, 1, N))
    y = np.sign(x)

    noise = np.random.rand(N) < p
    y[noise] *= -1
    return x, y

# calculate Ein
def calculate_Ein(x, y, s, theta):
    pred = s * np.sign(x-theta)
    return np.mean(pred != y)

# decision stump algorithm
def decision_stump(x, y):
    N = len(x)

    best_theta = None
    best_s = None
    min_Ein = float('inf')

    for i in range(N-1):
        theta = (x[i] + x[i+1]) / 2     # middle point
        for s in [-1, 1]:
            Ein = calculate_Ein(x, y, s, theta)
            if Ein < min_Ein:
                min_Ein = Ein
                best_theta = theta
                best_s = s

    return min_Ein, best_s, best_theta
```

Figure 11b. snapshot of my code

```
7ching@Mac hw2_src % python3 q11.py
Median of Eout(g) - Ein(g): 0.09504575620517976
```

Figure 11c. median of Eout(g) − Ein(g)

The scatter plot shows that $E_{in}(g)$ ranges from 0 to around 0.4, with most points clustered in the lower range. On the other hand, $E_{out}(g)$ ranges from 0 to almost 0.7. Here's what I think is happening:

Though the median is low, when $E_{in}(g)$ is very small, $E_{out}(g)$ still doesn't consistently stay low. This suggests that even if the model fits the training data well, it might not generalize well to new data. In cases where $E_{in}(g)$ approaches 0, the model seems to be overfitting. In some experiments, $E_{out}(g)$ shoots up to 0.7, showing poor generalization despite great training performance.

These results show that, especially with small datasets, training error isn't always a reliable predictor of test error, which is why the model might perform well on training data but still show a lot of variability on unseen data.

For problems 11 and 12, the full code has been made available on GitHub.
Feel free to check it out if required or interested!

12.

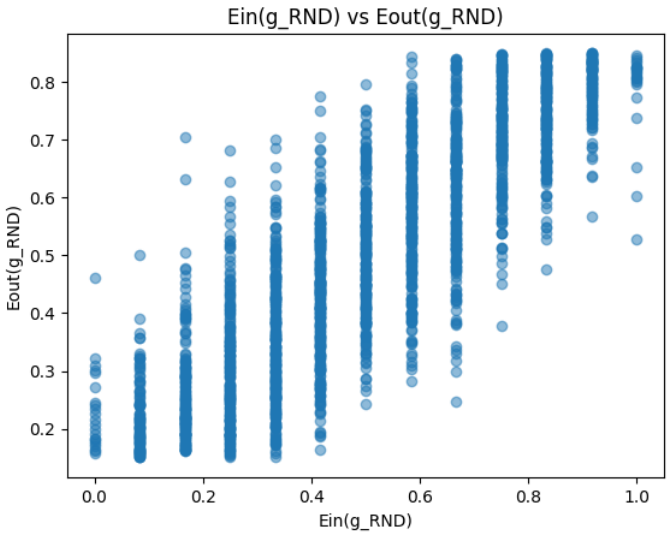

Figure 12a. scatter plot of (Ein(gRND), Eout(gRND))

```python
import numpy as np
import matplotlib.pyplot as plt

# generate datas
def generate_data(N,p):
    x = np.sort(np.random.uniform(-1, 1, N))
    y = np.sign(x)

    noise = np.random.rand(N) < p
    y[noise] *= -1
    return x, y

# calculate Ein
def calculate_Ein(x, y, s, theta):
    pred = s * np.sign(x-theta)
    return np.mean(pred != y)

# random hypothesis
def random_hypothesis():
    s = np.random.choice([-1, 1])
    theta = np.random.uniform(-1, 1)
    return s, theta

# calculate Eout
def calculate_Eout(s, theta, p):
    v = s * (0.5 - p)
    u = 0.5 - v
    return u + v * np.abs(theta)

# main code
Ein_list = []
Eout_list = []

TIMES = 2000    # 2000 times
N = 12          # data size: 12
P = 0.15        # p: 15%

for i in range(TIMES):
    x, y = generate_data(N, P)
```

Figure 12b. snapshot of my code

```
7ching@Mac hw2_src % python3 q12.py
Median of Eout(g_RND) - Ein(g_RND): 0.005474494316652034
```

Figure 12c. median of Eout(gRND) − Ein(gRND)

In this plot, $E_{in}(g_{RND})$ ranges from 0 to 1, while $E_{out}(g_{RND})$ falls between 0.2 and 0.8. Compared to the results from Problem 11, this plot shows a wider range. As $E_{in}(g_{RND})$ increases, $E_{out}(g_{RND})$ also tends to go up. Unlike in Problem 11, where we selected the best $E_{in}(g)$ hypothesis, randomly choosing s and θ leads to weaker generalization.

However, even when the median is extremely low, it doesn't necessarily result in good performance. This small median might indicate instability in both training and test performance when using a random hypothesis. So, this slightly difference is likely because both the training and test performance are poor, rather than showing good generalization.

In conclusion, carefully selecting the hypothesis is much more effective for ensuring that the model generalizes well, compared to the random approach.

For problems 11 and 12, the full code has been made available on GitHub.
Feel free to check it out if required or interested!

13.