CS/CNS/EE 156a

Homework 4

Sung Hoon Choi

(Last name: Choi)

1. Answer: [d]

$$\delta = 4m_H(2N)e^{-\frac{1}{8}\epsilon^2 N}$$

$$= 4(2N)^{d_{VC}}e^{-\frac{1}{8}\epsilon^2 N}$$

$$= 4(2N)^{10}e^{-\frac{1}{8}*0.05^2*N}$$

We want δ to be close to 0.05.

Choice a): For N=400,000: $\delta = 2.21 * 10^5$
Choice b): For N=420,000: $\delta = 697.75$
Choice c): For N=440,000: $\delta = 2.145$
Choice d): For N=460,000: $\delta = 0.006$
Choice e): For N=480,000: $\delta = 0.000019$
Therefore, if we want 95% confidence that our generalization error is at most 0.05, the number of samples needed is closest to choice [d].

2. Answer: [d]

$$\delta = 0.05\ d_{vc} = 50\ N = 10{,}000\ m_H = N^{d_{vc}} = N^{50}$$

a)

$$\epsilon \leq \sqrt{\frac{8}{N} ln \frac{4m_H(2N)}{\delta}}$$

$$= \sqrt{\frac{8}{N} ln \frac{4(2N)^{50}}{\delta}}$$

$$= \sqrt{\frac{8}{10000} \ln(\frac{4(2 * 10000)^{50}}{0.05})}$$

$$= 0.6322$$

b)

$$\epsilon \leq \sqrt{\frac{2\ln\left(2Nm_H(N)\right)}{N}} + \sqrt{\frac{2}{N} ln \frac{1}{\delta}} + \frac{1}{N}$$

$$= \sqrt{\frac{2\ln(2N * N^{50})}{N}} + \sqrt{\frac{2}{N} ln \frac{1}{\delta}} + \frac{1}{N}$$

$$= \sqrt{\frac{2\ln(2 * 10000 * 10000^{50})}{10000}} + \sqrt{\frac{2}{10000} \ln(\frac{1}{0.05})} + \frac{1}{10000}$$

$$= 0.3313$$

c)

$$\epsilon \leq \sqrt{\frac{1}{N}(2\epsilon + ln\frac{6m_H(2N)}{\delta})}$$

$$= \sqrt{\frac{1}{N}(2\epsilon + ln\frac{6 * (2N)^{50}}{\delta})}$$

$$= \sqrt{\frac{1}{10000}(2\epsilon + \ln(\frac{6 * (2 * 10000)^{50}}{0.05}))}$$

If we solve this inequality, we get

$$\epsilon \leq 0.2237$$

d)

$$\epsilon \leq \sqrt{\frac{1}{2N}(4\epsilon(1+\epsilon) + ln\frac{4m_H(N^2)}{\delta}}$$

$$= \sqrt{\frac{1}{2N}(4\epsilon(1+\epsilon) + ln\frac{4(N^2)^{50}}{\delta}}$$

$$= \sqrt{\frac{1}{2 * 10000}(4\epsilon(1+\epsilon) + \ln(\frac{4(10000^2)^{50}}{0.05}))}$$

If we solve this inequality, we get

$$\epsilon \leq 0.2152$$

Therefore, the smallest bound is [d].

3. Answer: [c]
We just repeat what we've done in problem2, but using N=5 this time.
Then, we get the bounds as following:

Choice a): 13.828 ($\epsilon \leq \sqrt{\frac{8}{5}\ln(\frac{4(2*5)^{50}}{0.05})} = 13.828$)

Choice b): 7.049 ($\epsilon \leq \sqrt{\frac{2\ln(2*5*5^{50})}{5}} + \sqrt{\frac{2}{5}\ln\left(\frac{1}{0.05}\right)} + \frac{1}{5} = 7.049$)

Choice c): 5.101 ($\epsilon \leq \sqrt{\frac{1}{5}(2\epsilon + \ln(\frac{6*(2*5)^{50}}{0.05}))}$ => $\epsilon \leq 5.101$)

Choice d): 5.593 ($\epsilon \leq \sqrt{\frac{1}{2*5}(4\epsilon(1+\epsilon) + \ln(\frac{4(5^2)^{50}}{0.05})}$ => $\epsilon \leq 5.593$)

Thus, the smallest bound is [c].

4. Answer: [e]

We can find the slope for the least mean square error by using the formula on lecture slide 3.

$$\text{W}=X^{\dagger}\text{y} \quad \text{while} \quad X^{\dagger} = (X^T X)^{-1} X^T$$

$$X^{\dagger} = \left((x_1\ x_2) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right)^{-1} (x_1\ x_2) = \frac{1}{{x_1}^2+{x_2}^2} (x_1\ x_2)$$

$$\text{Thus,}\ \ w_1 = \frac{1}{{x_1}^2+{x_2}^2} (x_1\ x_2) \begin{pmatrix} y1 \\ y2 \end{pmatrix} = \frac{x1y1+x2y2}{{x_1}^2+{x_2}^2}$$

By generating the slope 10000 times and taking its average, I got 1.43.

Since there is no matching option, the answer is [e].

Please see the code below for the actual derivation.

```
#Sung Hoon Choi
#CS/CNS/EE156a HW4 Problem 4
import math
import random

# The target sine function.
def f(x):
    return math.sin(math.pi * x)

# By lecture note 3, we can find the slope that minimizes mean square error as:
# w = x_psuedo_inverse * y
# If we solve this matrix arithmetic for two point cases (x1,y1) (x2,y2), then we obtain
# regression_slope = (x1y1+x2y2)/(x1^2+x2^2)
def regression_slope(x1, f_x1, x2, f_x2):
    return ((x1*f_x1)+(x2*f_x2))/(x1**2+x2**2)   # slope for the linear equation that minimizes mean square error.

# return the corresponding g_bar(x) = ax
def g(slope, x):
    return slope * x

total_repetition = 10000
total_slope = 0

for i in range (0, total_repetition):
    x1 = (1 if random.random() < 0.5 else -1) * random.random() # x1 is a random value in [-1,1]
    x2 = (1 if random.random() < 0.5 else -1) * random.random() # x2 is a random value in [-1,1]
    f_x1 = f(x1)
    f_x2 = f(x2)
    current_slope = regression_slope(x1, f_x1, x2, f_x2)
    total_slope = total_slope + current_slope

g_bar_slope = total_slope/total_repetition
print ("g_bar's slope: ", g_bar_slope)  #Answer for Problem 4.
```

5. Answer: [b]

The bias I got was 0.27.

Please see the code below for derivation.

```
#Sung Hoon Choi
#CS/CNS/EE156a HW4 Problem 5
import math
import random

# The target sine function.
def f(x):
    return math.sin(math.pi * x)

# By lecture note 3, we can find the slope that minimizes mean square error as:
# w = x_psuedo_inverse * y
# If we solve this matrix arithmetic for two point cases (x1,y1) (x2,y2), then we obtain
# regression_slope = (x1y1+x2y2)/(x1^2+x2^2)
def regression_slope(x1, f_x1, x2, f_x2):
    return ((x1*f_x1)+(x2*f_x2))/(x1**2+x2**2)   # slope for the linear equation that minimizes mean square error.

# return the corresponding g_bar(x) = ax
def g(slope, x):
    return slope * x
```

```
total_repetition = 10000
total_slope = 0

for i in range (0, total_repetition):
    x1 = (1 if random.random() < 0.5 else -1) * random.random() # x1 is a random value in [-1,1]
    x2 = (1 if random.random() < 0.5 else -1) * random.random() # x2 is a random value in [-1,1]
    f_x1 = f(x1)
    f_x2 = f(x2)
    current_slope = regression_slope(x1, f_x1, x2, f_x2)
    total_slope = total_slope + current_slope

g_bar_slope = total_slope/total_repetition
print ("g_bar's slope: ", g_bar_slope)  #Answer for Problem 4.

######################### Code for Problem 5 starts ############################

total_bias = 0
for i in range (0, total_repetition):
    x = (1 if random.random() < 0.5 else -1) * random.random() # x is a random value in [-1,1]
    current_bias = (g(g_bar_slope, x) - f(x))**2
    total_bias = total_bias + current_bias

print("bias: ", total_bias/total_repetition) #Answer for Problem 5
```

---

6. Answer: [a]

The variance I got was 0.23.

Please see the code below for derivation.

```
#Sung Hoon Choi
#CS/CNS/EE156a HW4 Problem 6
import math
import random

# The target sine function.
def f(x):
    return math.sin(math.pi * x)

# By lecture note 3, we can find the slope that minimizes mean square error as:
# w = x_psuedo_inverse * y
# If we solve this matrix arithmetic for two point cases (x1,y1) (x2,y2), then we obtain
# regression_slope = (x1y1+x2y2)/(x1^2+x2^2)
def regression_slope(x1, f_x1, x2, f_x2):
    return ((x1*f_x1)+(x2*f_x2))/(x1**2+x2**2)   # slope for the linear equation that minimizes mean square error.

# return the corresponding g_bar(x) = ax
def g(slope, x):
    return slope * x

total_repetition = 10000
total_slope = 0

for i in range (0, total_repetition):
    x1 = (1 if random.random() < 0.5 else -1) * random.random() # x1 is a random value in [-1,1]
    x2 = (1 if random.random() < 0.5 else -1) * random.random() # x2 is a random value in [-1,1]
    f_x1 = f(x1)
    f_x2 = f(x2)
    current_slope = regression_slope(x1, f_x1, x2, f_x2)
    total_slope = total_slope + current_slope

g_bar_slope = total_slope/total_repetition
print ("g_bar's slope: ", g_bar_slope)  #Answer for Problem 4.

######################### Code for Problem 6 starts ############################

total_variance = 0
for i in range(0, total_repetition):
    x1 = (1 if random.random() < 0.5 else -1) * random.random()  # x1 is a random value in [-1,1]
    x2 = (1 if random.random() < 0.5 else -1) * random.random()  # x2 is a random value in [-1,1]
    f_x1 = f(x1)
    f_x2 = f(x2)
    current_slope = regression_slope(x1, f_x1, x2, f_x2)

    D_x = (1 if random.random() < 0.5 else -1) * random.random()  # x2 is a random value in [-1,1]
    current_variance = (g(current_slope, D_x) - g(g_bar_slope,D_x))**2
```

```
    total_variance = total_variance + current_variance

print("variance: ", total_variance/total_repetition) # Answer for Problem 6
```

7. Answer: [b]
Choice a): By lecture note (slide 8, p.15), the expected E_{out} is 0.50 + 0.25 = 0.75
Choice b): By problem 5 and 6, the expected E_{out} is 0.27+0.23 = 0.5
Choice c): By lecture note (slide 8, p.15), the expected E_{out} is 0.21+1.69 = 1.90
Choice d) and e): By inspection, we can infer that square function hypotheses built from two examples would not perform well for a sine wave target function, when compared to linear hypotheses.
Therefore, choice [b] has the least expected value of out-sample error.

8. Answer: [c]

$$m_H(1) = 2$$

$$m_H(2) = 2m_H(1) - \binom{1}{q} = 2m_H(1) = 2^2 \ (q > 1)$$

$$m_H(3) = 2m_H(2) - \binom{2}{q} = 2m_H(2) = 2^3 \ (q > 2)$$

…

$$m_H(q) = 2m_H(q-1) - \binom{q-1}{q} = 2m_H(q-1) = 2^q$$

$$m_H(q+1) = 2m_H(q) - \binom{q}{q} = 2^{q+1} - 1$$

Thus, as we can see, q is the largest value of N for which $m_H(N) = 2^N$. When it's q+1, the $m_H(q+1)$ is $2^{q+1} - 1$. Therefore, the VC dimension of the hypothesis is q, and the answer is [c].

9. Answer: [b]
If we take the intersection of hypothesis sets, the intersection would not be able to shatter more points than the largest number of points that the hypothesis with the minimum VC dimension (out of the given hypotheses set) could shatter. Besides, VC dimensions by definition cannot be negative. Therefore, the answer is [b].

10. Answer: [d] or [e]
If we take the union of hypothesis set, the union would be able to shatter more points than the [VC dimension] number of points from the hypothesis that has the largest VC dimension. Besides, since a "union" contains overlaps, the union would definitely be smaller than the total summation. Unfortunately, I don't know where "K-1" term in choice [e] comes from. Therefore, I would hedge by selecting choice [d] and choice [e].