Machine Learning Foundations HW2

b05901060 賴繹文

1. c
    a. There are three points in a line, which can't be shattered if the middle point has different sign with others,
    b. There are three points in a line and another point. Same as (a), (b) can't be shattered.
    c. There are two points in a line, and two other points which can't form a plane with the line. Thus, any combination of inputs can be separated by a plane.
    d. Two lines with two points on it. These two lines can form a plane and thus four points can't be shattered by a plane.
    e. There are four points in a line and another point. Same as (a), it can't be shattered.
2. d, 4n-2.
   Consider the max dichotomies of patterns from 1 positive to N positive, then multiplied by two, which is the answer. On a 2D plane, there will be N choices of horizontal or vertical line, respectively. Thus, 2N patterns of inputs are shown. While the case that all N points are positive, the horizontal one and the vertical one should be calculated once. So there are 2N - 1 patterns. After multiplying 2, 4N - 2 is the result. Mentioned that 2N - 1 is the maximum case, which means that it is possible for other cases to exist.
3. c
   First consider the case "2 inputs" with different signs. There are infinite numbers of lines to separate two points with different slopes. And given a w0 > 0, we can always find a line matching the conditions to classify two points, by shifting or rotating the line. 2 inputs with the same sign is the much easier case, which is also found in a line to classify. Thus, "2 inputs case" can be shattered. Further, consider the case with three inputs and three aren't on the same line. Based on the discussion with case "two inputs", the procedure shifting or rotating the line can't make sure that the other one point to be classified as the same sign. In other words, to meet the condition w0 >0, this case can't be classified correctly, same as the case "three inputs on the same line". As a result, there is no case shattered when there are three inputs, so the VC dimension is 2.
4. b
   Same as the case "Positive Intervals". Calculate all the distances to (0,0,0), and map the value to a 1D line. Select two from N+1 spots and plus one for a = b = 0.
5. b
   Same as "Positive Intervals", its VC dimension is 2. If there are three points on the same line and the middle one with different signs with others. This case can't be shattered.
6. d

$$P[E_{in}(g) - E_{out}(g) > \varepsilon] < 4\, m_H(2N) \exp\left(-\frac{1}{8}\varepsilon^2 N\right)$$

$$P[E_{out}(g^*) - E_{in}(g^*) > \varepsilon] < 4\, m_H(2N) \exp\left(-\frac{1}{8}\varepsilon^2 N\right)$$

$$E_{out}(g) \le E_{in}(g) + \sqrt{\frac{8}{N} \ln\left(\frac{4m(2N)}{\delta}\right)}$$

$$E_{in}(g^*) \le E_{out}(g^*) + \sqrt{\frac{8}{N} \ln\left(\frac{4m(2N)}{\delta}\right)}$$

$$E_{out}(g) - E_{out}(g^*) \le \underbrace{E_{in}(g) - E_{in}(g^*)}_{<0} + 2\sqrt{\frac{8}{N} \ln\left(\frac{4m(2N)}{\delta}\right)}$$

Here we assume that both data outside or inside the training area are large enough.The second line means that we change the character of testing data and the training data. At the last line, there is a term less than 0, which means that there is an upper bound, the latter term.

7. d

   $2^N = M$ is the case with the maximum number hypothesis sets. N is the number of the inputs, which is also VC dimension. $N = log_2 M$

8. d

   From zero of -1 to all numbers are -1, there are k+1 patterns. Thus VC dimension is k+1.

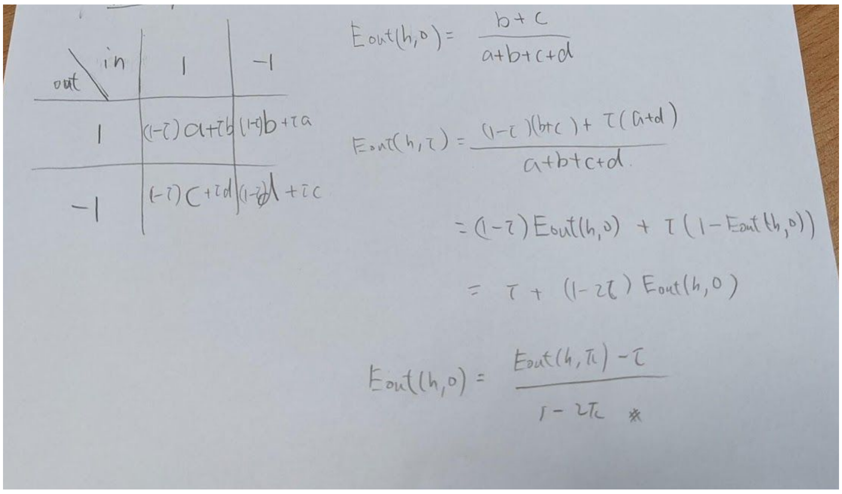
9. c

   “some set of d distinct inputs is shattered by H”, “some set of d + 1 distinct inputs is not shattered by H”, and “any set of d + 1 distinct inputs is not shattered by H”. The first one is by the definition of break point mentioned in lecture 5, p24. The second and third one are the same as what’s in lecture 7 p13. It provide the condition that $d_{VC} <= d+1$.

10. c

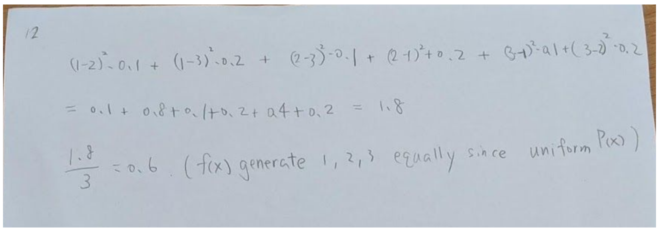
    Apply the concept of Polar coordinate system, and the sine value stands for the imaginary part. Given all the inputs with different signs, we can map those to a circle on the polar coordinate system. There are different points with different signs in the circle. Taught in class, this case is a “convex set”. Any combination of inputs can be shattered by a convex region. Thus the VC dimension of c is infinite.

11. d



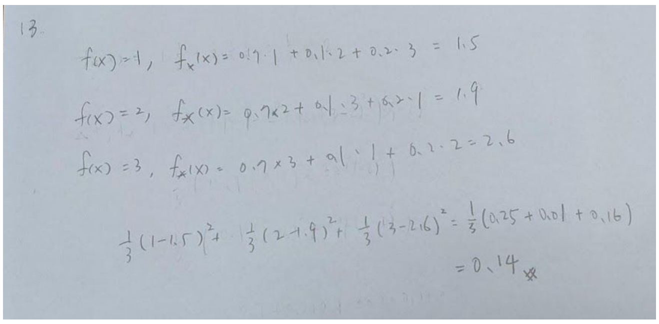
There are four cases on the left side of the picture, considering the hypothesis and the ground truth on out-sample data. When flipping the ground truth answer with probability, the amount of four cases will change, shown in figure. By the basic definition of expected error, we can conclude the relationship between two kinds of error. (The left and top of the table should be corrected, which is hypothesis and ground truth.)

12. b



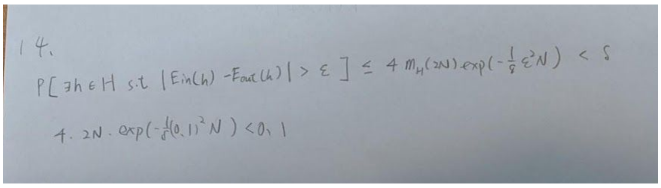
There are six kinds of errors. 1,2,3, three outcomes of f(x) will be changed to the other 2 numbers with probability. Thus, 3 multiplied by 2 is six. Using the sum of squared error multiplied by its corresponding probability, we got the answer.

13. b



Calculate the f star of x with the three outcomes first. And calculate the probability of different y outcomes. Surprisingly, the probability of 1,2,3 is also one third. Take "y = 1" for example, 0.7 from f(x) = 1, 0.1 from f(x) = 3, and 0.2 from f(x) = 2. And we can calculate the squared difference of f and f*.

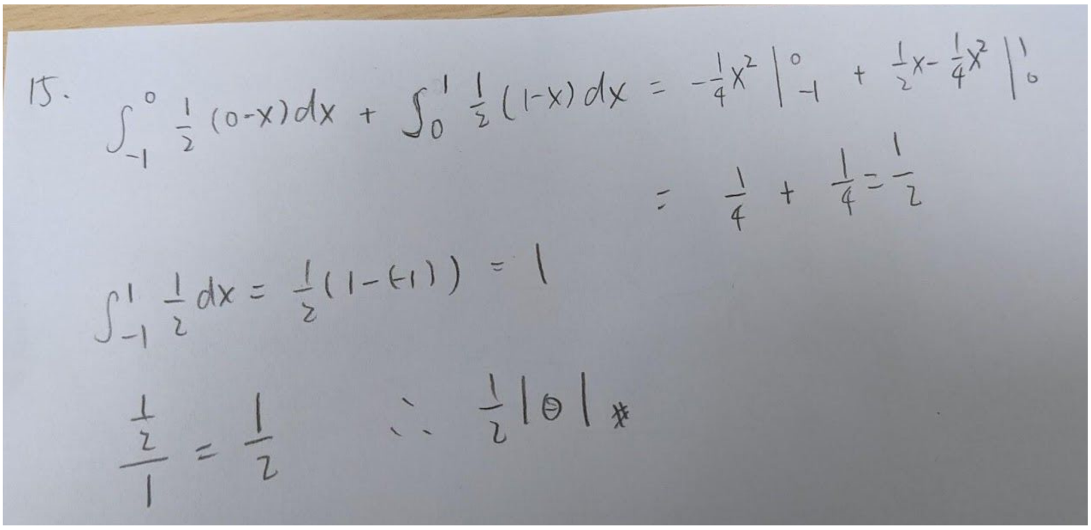
14. d



Applying the formula of (6), we get the inequality in line two. Then solve it by Wolfram, the answer is shown below.
N > 11000(approximately)

Consider the smallest answer in all choices but also in the range of solution, (2),12000 is the answer.

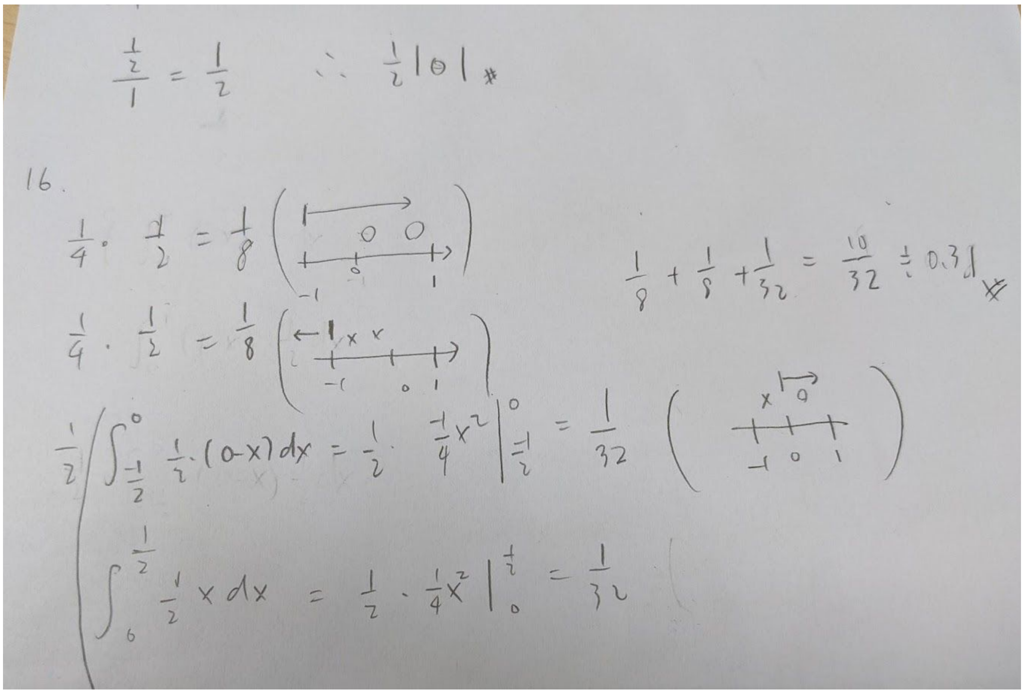
15. b
From the figure below, we can see that the expected error of positive rays can be splitted to two parts. The first part is rays start point at (-1,0), and the other one is at (0,1). The first term in integration is the probability of a point under continuous uniform distribution, and the second term is the error amount. Finally, the ratio of expected error to rectangle area of interval is 0.5, which means that half of samples would be the error.



```
yiwenlai@YiWens-MacBook-Pro Downloads % python3 hw2_full.gyp
size  2  flip  0 Eout-Ein  0.28529189999999993
size  2  flip  0.1 Eout-Ein  0.3653248
size  20  flip  0 Eout-Ein  0.023817099999999973
size  20  flip  0.1 Eout-Ein  0.04940839999999994
size  200  flip  0 Eout-Ein  0.002426699999999998
size  200  flip  0.1 Eout-Ein  0.005594099999999988
yiwenlai@YiWens-MacBook-Pro Downloads % 
```

figure A

16. d, size = 2, flip = 0 in figure A
    a. Analytical proof
    Consider three cases after sorted, with probability 0.25,0.25,0.5. Multiply each probability with the expected error value. In the first case, the number less than 0 would be considered as positive; in the second case, the number greater than 0 would be considered as negative; In the third case, the value of theta will fall in (-0.5,0.5) equally. We can separate it as two cases, theta in (-0.5,0) and (0,0.5). Then calculate the expected error. Sum of those value is the answer.



17. b, size = 20, flip = 0 in figure A
18. e, size = 2, flip = 0.1 in figure A
19. c, size = 20, flip = 0.1 in figure A
20. a, size = 200, flip = 0.1 in figure A

Code

```
import random
import numpy as np

def sign(x):
    if x > 0:
        return 1
    else:
        return -1
def theta_generator(x):
    theta_list = []
    theta_list.append(-1)
    for i in range(len(x)-1):
        if x[i] != x[i+1]:
            theta_list.append((x[i] + x[i+1])/2)
    return theta_list
def label_generator(size,tau):
    x = np.random.uniform(0,1,size)
    y = []
    for i in x:
        if i < tau:
            y.append(-1)
        else:
            y.append(+1)
    return y
def algorithm(x,tau):
    x = sorted(x)
    theta_list = theta_generator(x)
    y_list = label_generator(len(x),tau)
    S = 99999
    Theta = 99999
    Err = 99999
    for s in [-1,1]:
        for theta in theta_list:
```

```
            err__count = 0
            for j in range(len(x)):
                if  (y_list[j] * sign(x[j])) !=
(sign(x[j]-theta) * s): #
                    err_count += 1
            if err_count < Err :
                Err = err_count
                S = s
                Theta = theta
            elif err_count == Err:
                if s + theta < S + Theta:
                    S = s
                    Theta = theta
   return S, Theta, Err/(len(x))
# main
for size in [2,20,200]:
   for tau in [0,0.1]:
       diff = []
       for i in range(10000):
           x_in = np.random.uniform(-1,1,size)
           x_in = list(x_in)
           s,theta,e_in = algorithm(x_in,tau)

           e_out = 0
           x_out = np.random.uniform(-1,1,100000)
           x_out = list(x_out)
           x_out = sorted(x_out)

           y_out_label = label_generator(len(x_out),tau)

           for k in range(len(x_out)):
               if y_out_label[k] * sign(x_out[k]) !=
(sign(x_out[k] - theta) * s):#
                   e_out += 1
```

```
            diff.append(e_out/len(x_out) - e_in )
        print("size ", size, " flip ", tau, "Eout-Ein
",sum(diff)/len(diff))
```