Homework 2

1. (d)

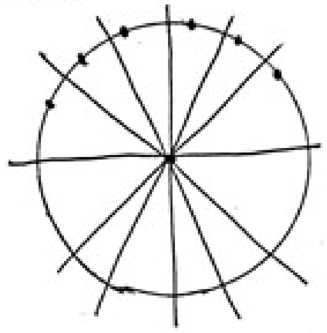
Q1 Prove:

We need to find the maximum dichotomies of all possible data sets

1. We can normalize all vectors from the lucky-point O to x_i, and transform it into N dots on a unit circle with centre O.
2. It is easy to see that if there are dots that are on the same line crossing O, the dichotomies will be lesser, since they can only be the same sign or only be different sign.
3. Consider the other cases, we can always seperate N dots into two half, A dots and B dots. ① By first considering the A dots it is easy to see that there are $2 \cdot A$ dichotomies. ② Next for the other half every points in B, we can draw a line from the dot crossing O and by rotating the line with a tiny angle up and down we can create 2 new dichotomies for every dots in B.

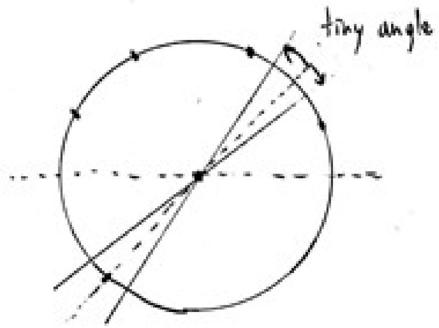
   By adding up $2A + 2B = 2N$, we know that the maximum dichotomies are $2N$, so $m_H(N) = 2N$

Illustration:



6 dots in A

12 dichotomies



2 new dichotomies

2. (a)
   Prove: There is two upper bound for d_vc, one is depend on the dimension of x which is d_vc <= 6210 + 1, and the other one is depend on the number of perceptrons, since one perceptron can implement one dichotomy, we can use 2^N <= 1126 to find out d_vc <= lg(1126). And it is obvious that d_vc <= lg(1126) is a tighter bound.
3. (e)
   Prove:
   (a) d_vc = 4, since for N = 5, we can't implement {1, -1, 1, -1, 1} this dichotomy.
   (b) d_vc = 4, since mh(5) < 2^5, because {1, -1, 1, -1, 1} and {-1, 1, -1, 1, -1} are both not dichotomies.
   (c) d_vc = infinity, since sin is a oscillation function by putting N points properly we can implement 2^N dichotomies by changing different w.
   (d) d_vc = 4, since no set of 5 points can be fully shattered. By similar labeling in (e) we can label four extreme point plus one inside the four. Then it is impossible to label the point inside negative while the extreme points is labeled positively.
   (e) d_vc = 3, since no set of 4 points can be fully shattered. We can label four extreme points with P_top, P_down, P_left, P_right. Assuming P_top_y - P_down_y > P_right_x - P_left_x, then P_top and P_down can't be labeled positively while P_left and P_right are labeled negatively.
4. (b)

Q4 Prove:

① prove $d_{vc} \le 2M$

By given 2M+1 inputs we can sort the input according to its $x^T x$,
we can see that each a_i, b_i adds a positive interval, so we have M intervals
And for $\{x_1^T x_1, x_2^T x_2, \dots, x_{2M+1}^T x_{2M+1}\}$, it is impossible to create $\underbrace{\{1,-1,1,\dots,-1,1\}}_{2M+1}$ with M intervals.

② prove $d_{vc} \ge 2M$

Let A be an arbitrary set, with $a_i \in \{-1,1\}$.
We can use one interval for each block of adjacent 1s in A.
And for $|A| = 2M$, there is at most M isolated 1s, and thus, we need at most M intervals.

5. (b)
   - Some set of d + 1 distinct inputs is not shattered by H
   - Any set of d + 1 distinct inputs is not shattered by H
6. (b)

Q6. Prove:

$$E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N}(wx_n - y_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n^2 \cdot w^2 - 2 \cdot x_n \cdot y_n \cdot w + y_n^2)$$

$$\nabla E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N} 2 \cdot x_n^2 \cdot w - 2 \cdot x_n \cdot y_n)$$

$$= \frac{1}{N}\left(2 \cdot w \cdot \sum_{n=1}^{N} x_n^2 - 2 \sum_{n=1}^{N} x_n \cdot y_n\right) = 0$$

$$\Rightarrow w = \frac{\sum_{n=1}^{N} y_n \cdot x_n}{\sum_{n=1}^{N} x_n^2}$$

7. (e)
   Explain:
   (a) for poisson distribution, it's mean = λ, so x_bar is the maximum likelihood estimate of λ
   (b) for Gaussian distribution, it's mean = μ, so x_bar is the maximum likelihood estimate of μ
   (c) for Laplace distribution, it's mean = μ, so x_bar is the maximum likelihood estimate of μ
   (d) for geometric distribution, it's mean = 1/θ, so 1/x_bar is the maximum likelihood estimate of θ
   Hence every choice is correct.

8. (a)

Q8. Derivation:

$$\max_{\tilde{h}} \prod_{n=1}^{N} \tilde{h}(y_n \cdot x_n) \Rightarrow \min_{w} \frac{1}{N} \sum_{n=1}^{N} -\ln\left(\frac{1+y_n \cdot w^T \cdot x_n + |y_n \cdot w^T \cdot x_n|}{2+2|y_n \cdot w^T x_n|}\right)$$

$$\min_{w} \tilde{E}_{in}(w) = \frac{-1}{N} \sum_{n=1}^{N} \ln\left(1+y_n \cdot w^T x_n + |y_n \cdot w^T \cdot x_n|\right) - \ln\left(2+2|y_n \cdot w^T \cdot x_n|\right)$$

for $y_n \cdot w^T \cdot x_n \geq 0$:

$$\frac{\partial \tilde{E}_{in}(w)}{\partial w_i} = \frac{-1}{N} \sum_{n=1}^{N} \frac{2 y_n \cdot x_{n,i}}{1+y_n \cdot w^T \cdot x_n + |y_n \cdot w^T \cdot x_n|} - \frac{2 y_n \cdot x_{n,i}}{2+2|y_n \cdot w^T \cdot x_n|}$$

$$= \frac{-1}{N} \sum_{n=1}^{N} \frac{y_n \cdot x_{n,i}}{(1+y_n w^T x_n + |y_n \cdot w^T x_n|) \cdot (1+|y_n \cdot w^T \cdot x_n|)}$$

Similar for $y_n \cdot w^T \cdot x_n \leq 0$:

$$\frac{\partial \tilde{E}_{in}(w)}{\partial w_i} = \frac{-1}{N} \sum_{n=1}^{N} \frac{y_n \cdot x_{n,i}}{(1+y_n \cdot w^T x_n + |y_n \cdot w^T x_n|) \cdot (1+|y_n \cdot w^T x_n|)}$$

//

$$\nabla \tilde{E}_{in}(w) = \frac{-1}{N} \sum_{n=1}^{N} \frac{y_n \cdot x_n}{(1+y_n \cdot w^T \cdot x_n + |y_n \cdot w^T \cdot x_n|) \cdot (1+|y_n \cdot w^T x_n|)}$$

9. (b)

Q9 Derivation:

for linear regression $\nabla E_{in}(w) = \frac{2}{N}(X^T X w - X^T y)$

Hessian matrix $= \nabla^2 E_{in}(w) = \frac{2}{N} X^T X$

10. (a)

Q10 Derivation:

$$\nabla E_{in}(w^*) = \frac{2}{N}(X^T X w^* - X^T y) = 0 \Rightarrow w^* = (X^T X)^{-1} X^T y$$

$$w_1 \leftarrow w_0 - \left(\frac{2}{N} X^T X\right)^{-1} \cdot \left(\frac{2}{N} X^T X w_0 - \frac{2}{N} X^T y\right)$$

$$w_1 = (X^T X)^{-1} X^T y \Rightarrow w_1 = w^*$$

11. (d)

Q11 Explain:

$$P[|E_{in}(g)-E_{out}(g)|>\epsilon] \le 4\cdot(2N)^{d_{vc}} e^{(-\frac{1}{8}\epsilon^2\cdot N)}$$

for $d_{vc}=2$, $\epsilon=0.05$, $\delta=0.1$

$$4(2N)^2\cdot e^{(-\frac{1}{8}\cdot 0.05^2\cdot N)} \le 0.1 \Rightarrow N \ge 89000 \text{ approximately}$$

12. (d)

Q12 Prove:

for $0\le|\theta|\le 0.5$, $E_{out}(h,r) = |\theta|\cdot(1-\tau) + (1-|\theta|)\cdot\tau$

$$= |\theta|\cdot(1-2\tau)+\tau$$

for $|\theta|>0.5$, $E_{out}(h,r) = \frac{1}{2}\cdot(1-\tau)+\frac{1}{2}\cdot\tau$

$$= \frac{1}{2}$$

$$E_{out}(h,r) = \min(|\theta|,0.5)\cdot(1-2\tau)+\tau$$

13. (b)
14. (b)
15. (c)
16. (b)

```
import numpy as np
import random

size = 2    #change this for different size
test = 10000
r = 0    #change this for different r
x_array = np.zeros(size)
y_array = np.zeros(size)
theta_array = np.zeros(size)
theta_array[0] = -0.5    #for -inf
s_array = np.array([-1,1])
E_out_minus_E_in = np.zeros(test)
for i in range(test):
    for j in range(size):
        x = random.uniform(-0.5, 0.5)
        x_array[j] = x
    x_array = np.sort(x_array)
    for j in range(size):
        rand = random.uniform(0,1)
        if rand < r:
            if x_array[j] > 0:
                y_array[j] = -1
            else:
                y_array[j] = 1
        else:
            if x_array[j] > 0:
                y_array[j] = 1
            else:
                y_array[j] = -1
    for j in range(size-1):
        theta_array[j+1] = (x_array[j] + x_array[j+1]) / 2
    opt_s = 0
    opt_theta = 0
    E_in = 1
    for j in range(2):
        wrong_data = 0
        for k in range(size):
            if y_array[k] != s_array[j]:
                wrong_data += 1
        for k in range(size):
            tmp = wrong_data / size
            if tmp < E_in:
                E_in = tmp
                opt_s = s_array[j]
                opt_theta = theta_array[k]
            if tmp == E_in and s_array[j] * theta_array[k] < opt_s * opt_theta:
```

```
                E_in = tmp
                opt_s = s_array[j]
                opt_theta = theta_array[k]
            if y_array[k] == s_array[j]:
                wrong_data += 1
            else:
                wrong_data -= 1
    E_out = abs(opt_theta) * (1-2*r) + r
    E_out_minus_E_in[i] = E_out - E_in

print(np.mean(E_out_minus_E_in))
```

17. (c)
18. (e)
19. (d)
20. (b)

```
import numpy as np
import random

filename_1 = 'data.txt'
filename_2 = 'data-test.txt'

inf = 1000000
s_array = np.array([1, -1])

train_size = 192
train_array = np.ones([train_size, 11])
with open(filename_1) as file:
    a = 0
    for line in file:
        line = line.strip().split()
        for i in range(11):
            train_array[a][i] = line[i]
        a += 1
file.close()

test_size = 64
test_array = np.ones([test_size, 11])
with open(filename_2) as file:
    a = 0
    for line in file:
        line = line.strip().split()
        for i in range(11):
            test_array[a][i] = line[i]
        a += 1
file.close()

opt_s = 0
opt_theta = 0
opt_i = 0
E_in = 1
opt_s_wob = 0
opt_theta_wob = 0
opt_i_wob = 0
E_in_wob = 0
opt_s_wob_2 = 0
opt_theta_wob_2 = 0
opt_i_wob_2 = 0
theta_array = np.zeros(train_size)
theta_array[0] = -inf
for i in range(10):
    tmp_wob = 1
```

```
    train_array = train_array[train_array[:, i].argsort()]
    for j in range(train_size-1):
        theta_array[j+1] = (train_array[j][i] + train_array[j+1][i]) / 2
    for j in range(2):
        wrong_data = 0
        for k in range(train_size):
            if train_array[k][10] != s_array[j]:
                wrong_data += 1
        for k in range(train_size):
            tmp = wrong_data / train_size
            if tmp < E_in:
                E_in = tmp
                opt_theta = theta_array[k]
                opt_s = s_array[j]
                opt_i = i
            elif tmp == E_in and s_array[j] * theta_array[k] < opt_s * opt_theta:
                E_in = tmp
                opt_theta = theta_array[k]
                opt_s = s_array[j]
                opt_i = i
            if tmp < tmp_wob:
                tmp_wob = tmp
                opt_theta_wob_2 = theta_array[k]
                opt_s_wob_2 = s_array[j]
                opt_i_wob_2 = i
            elif tmp == tmp_wob and s_array[j] * theta_array[k] < opt_s_wob_2 *
opt_theta_wob_2:
                tmp_wob = tmp
                opt_theta_wob_2 = theta_array[k]
                opt_s_wob_2 = s_array[j]
                opt_i_wob_2 = i
            if train_array[k][10] == s_array[j]:
                wrong_data += 1
            else:
                wrong_data -= 1
        if tmp_wob > E_in_wob:
            E_in_wob = tmp_wob
            opt_i_wob = opt_i_wob_2
            opt_s_wob = opt_s_wob_2
            opt_theta_wob = opt_theta_wob_2
print(E_in)    #Q17
print(E_in_wob)
print(E_in_wob - E_in)     #Q19

test_array = test_array[test_array[:, opt_i].argsort()]
wrong_data = 0
wrong_data_wob = 0
for i in range(test_size):
    if (opt_s * (test_array[i][opt_i] - opt_theta) > 0 and test_array[i][10] ==
-1) or (opt_s * (test_array[i][opt_i] - opt_theta) <= 0 and test_array[i][10] ==
1):
        wrong_data += 1
    if (opt_s_wob * (test_array[i][opt_i_wob] - opt_theta_wob) > 0 and
test_array[i][10] == -1) or (opt_s_wob * (test_array[i][opt_i_wob] -
```

```
opt_theta_wob) <= 0 and test_array[i][10] == 1):
        wrong_data_wob += 1
E_out = wrong_data / test_size
E_out_wob = wrong_data_wob / test_size
print(E_out)    #Q18
print(E_out_wob)
print(E_out_wob - E_out)    #Q20
```