Homework 1

1. (a)
   Explain: Tasks that's suited for machine learning have the following feature.

    1. Some underlying pattern to be learned ------ (b) doesn't have underlying pattern to be learned
    2. no programmable (easy) definition ------ (c) have programmable definition
    3. there is data about the pattern ------ (d) doesn't have the data about Zeus actual image

   (a) have all three feature so it's best suited for machine learning.

2. (d)

Prove Q_2:

To let w_{t+1}^T be correct on $(x_{n(t)}, y_{n(t)})$,

$sign(w_{t+1}^T \cdot x_{n(t)}) = y_n(t) \iff y_{n(t)} \cdot w_{t+1}^T \cdot x_{n(t)} > 0$

Let $w_{t+1} \leftarrow w_t + y_{n(t)} \cdot x_{n(t)} \cdot a$, $a \in R$

$y_{n(t)} \cdot w_{t+1}^T \cdot x_{n(t)} = y_{n(t)} \cdot w_t^T \cdot x_{n(t)} + a \cdot |x_{n(t)}|^2 > 0$

$\Rightarrow a > \frac{-y_{n(t)} \cdot w_t^T \cdot x_{n(t)}}{|x_{n(t)}|^2}$, since $\left\lfloor \frac{-y_{n(t)} \cdot w_t^T \cdot x_{n(t)}}{|x_{n(t)}|^2} + 1 \right\rfloor > \frac{-y_{n(t)} \cdot w_t^T \cdot x_{n(t)}}{|x_{n(t)}|^2}$

(d) is the correct answer

3. (c)

Prove Q_3:

By preprocessing $W_{t+1} \leftarrow W_t + y_n \cdot \frac{x_n}{|x_n|}$

Assume exists perfect W_f such that $y_n = \text{sign}(W_f^T x_n)$, $|W_f| = 1$

①

$$W_f^T \cdot W_{t+1} = W_f^T \left(W_t + y_n \cdot \frac{x_n}{|x_n|}\right)$$
$$\geq W_f^T \cdot W_t + \underbrace{\min_n y_n \cdot W_f^T \cdot \frac{x_n}{|x_n|}}_{} \rightarrow \rho_z = \min_n \frac{y_n \cdot W_f^T \cdot \frac{x_n}{|x_n|}}{|W_f|}$$

②

$$|W_{t+1}|^2 = \left|W_t + y_n \cdot \frac{x_n}{|x_n|}\right|^2$$
$$= W_t^2 + 2 \cdot y_n \cdot W_t^T \cdot \frac{x_n}{|x_n|} + 1, \quad \text{since } W_t \text{ only change when mistake } y_n \cdot W_t^T \cdot \frac{x_n}{|x_n|} \leq 0$$
$$\leq W_t^2 + 1$$

By ①, ② and $\cos\theta \leq 1 \Rightarrow 1 \geq \frac{W_f^T \cdot W_{t+1}}{|W_f| \cdot |W_{t+1}|} \geq \frac{T \cdot \rho_z}{1 \cdot \sqrt{T}}$

$$\Rightarrow T \leq \frac{1}{\rho_z^2}$$

4. (b)

Prove Q_4:

$$U_{orig} = \left(\frac{R}{\rho}\right)^2 = \frac{\max_n |x_n|^2}{(\min_n y_n W_f^T x_n)^2} \xrightarrow[n \text{ for } \min_n \text{ is } b]{\text{assume } n \text{ for } \max_n \text{ is } a} \frac{|x_a|^2}{(W_f^T x_b)^2}$$

$$U = \frac{1}{\rho_z^2} = \frac{|W_f|^2}{(\min_n y_n W_f^T \cdot \frac{x_n}{|x_n|})^2} \xrightarrow{\text{assume } n \text{ for } \min_n \text{ is } b} \frac{|x_b|^2}{(W_f^T \cdot x_b)^2}$$

Since we know x_a is the longest x_n when updating and x_b is the one that $W_f^T \cdot x_b$ closest to zero, there is a chance where $x_a = x_b$, so $U \leq U_{orig}$

5. (c)

PLA:

$$W_{t+1} \leftarrow W_t + y_{n(t)} \cdot x_{n(t)} \quad \text{, when } y_{n(t)} \cdot w_t^T \cdot x_{n(t)} < 0$$

Train:

$$W_1 = W_0 + (-1) \cdot \begin{bmatrix} 1 \\ -2 \\ 2 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ -2 \end{bmatrix}$$

check $-1 \cdot [-1,2,-2] \cdot \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} = 7 > 0$

check $1 \cdot [-1,2,-2] \cdot \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} = 3 > 0$

check $-1 \cdot [-1,2,-2] \cdot \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} = 3 > 0$

check $1 \cdot [-1,2,-2] \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = -1 < 0$

$$W_2 = \begin{bmatrix} -1 \\ 2 \\ -2 \end{bmatrix} + 1 \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ -1 \end{bmatrix}$$

Test:

$1 \cdot [0\ 3\ -1] \begin{bmatrix} 1 \\ \frac{1}{2} \\ 2 \end{bmatrix} = -\frac{1}{2} < 0$

$1 \cdot [0\ 3\ -1] \begin{bmatrix} 1 \\ \frac{1}{4} \\ 1 \end{bmatrix} = -\frac{1}{4} < 0$

$1 \cdot [0\ 3\ -1] \begin{bmatrix} 1 \\ \frac{1}{2} \\ 0 \end{bmatrix} = \frac{3}{2} > 0$

$-1 \cdot [0\ 3\ -1] \begin{bmatrix} 1 \\ -\frac{1}{2} \\ 1 \end{bmatrix} = \frac{5}{2} > 0$

PAM:

update if $y_n W_t^T x_n < 5$

Train:

$$W_1 = \begin{bmatrix} -1 \\ 2 \\ -2 \end{bmatrix}$$

check $-1 \cdot [-1,2,-2] \cdot \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} = 7 > 5$

check $1 \cdot [-1,2,-2] \cdot \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} = 3 < 5$

$$W_2 = \begin{bmatrix} -1 \\ 2 \\ -2 \end{bmatrix} + 1 \cdot \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ -2 \end{bmatrix}$$

check $-1 \cdot [0,4,-2] \cdot \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} = 4 < 5$

$$W_3 = \begin{bmatrix} 0 \\ 4 \\ -2 \end{bmatrix} + (-1) \cdot \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 5 \\ -2 \end{bmatrix}$$

check $1 \cdot [-1,5,-2] \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 2 < 5$

$$W_4 = \begin{bmatrix} -1 \\ 5 \\ -2 \end{bmatrix} + 1 \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \\ -1 \end{bmatrix}$$

Test:

$1 \cdot [0\ 6\ -1] \cdot \begin{bmatrix} 1 \\ \frac{1}{2} \\ 2 \end{bmatrix} = 1 > 0$

$1 \cdot [0\ 6\ -1] \cdot \begin{bmatrix} 1 \\ \frac{1}{4} \\ 1 \end{bmatrix} = \frac{1}{2} > 0$

$1 \cdot [0\ 6\ -1] \cdot \begin{bmatrix} 1 \\ \frac{1}{2} \\ 0 \end{bmatrix} = 3 > 0$

$-1\ [0\ 6\ -1] \cdot \begin{bmatrix} 1 \\ -\frac{1}{2} \\ 1 \end{bmatrix} = 4 > 0$

6. (a)

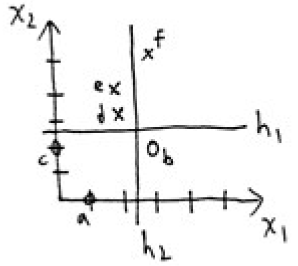
Explain: Because recommender problem needs to learn from real number rating which is a continuous value by some known rating data, so supervised regression best describes the associated learning problem.

7. (b)

   Explain: Because the labeler splits the outputs into two classes, the machine learns to categorizes new observations into one of two classes, so binary classification best describes the associated learning problem.

8. (e)

Explain Q_8:



① by picking b,c,d and $h_1((x_1,x_2)) = sign(2.5 - x_2)$

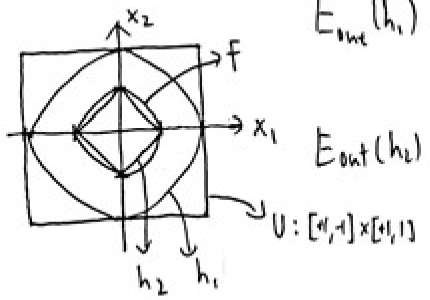
$E_{ots}(h_1) = \frac{0}{3} = 0$

② by picking b,d,e and $h_2((x_1,x_2)) = sign(x_1 - 2.5)$

$E_{ots}(h_2) = \frac{3}{3} = 1$

9. (d)

Derivation Q_9:



$$E_{out}(h_1) = \frac{1}{V}\cdot\sum_{(x,y)\in V}[h_1(x)\neq f(x)] = \frac{1}{4}\cdot\left(1^2\cdot\pi - \left(\tfrac{1}{2}\right)^2\cdot\pi\right) = \frac{3\pi}{16}$$

$$E_{out}(h_2) = \frac{1}{V}\cdot\sum_{(x,y)\in V}[h_2(x)\neq f(x)] = \frac{1}{4}\cdot\left(\left(\tfrac{1}{2}\right)^2\pi - \left(\tfrac{\sqrt{2}}{2}\right)^2\right) = \frac{\pi-2}{16}$$

10. (b)

Explain Q_{10}:

For the image above $Prob(h_1(x)=h_2(x)=f(x)) = \frac{\left(\frac{\sqrt{2}}{2}\right)^2 + (4-\pi)}{4} = \frac{9-2\pi}{8}$

$\left(\frac{9-2\pi}{8}\right)^4 \doteq 0.01336 \doteq 0.01$

11. (a)

Derivation Q_{11}:

Hoeffding's Inequality : $P[|\nu-\mu|>\epsilon] \le 2e^{(-2\epsilon^2 N)} \le 1-0.99$

$\Rightarrow 2\cdot e^{(-0.0002\cdot N)} \le 0.01 \Rightarrow N \ge 26491.58683\ldots$

12. (d)

Prove Q12:

By Hoeffding's Inequality we know $P[|\frac{C_m}{N} - P_m| > r] \le 2e^{(-2r^2N)}$

For the algorithm, $\underbrace{P[\bigcup_{i}^{M} |\frac{C_i}{N} - P_i| > r]}_{\text{is less than the probability that algo did not return } \epsilon\text{-optimal box}} \le 2 \cdot M \cdot e^{(-2r^2N)} \le \delta$

We need to find the relation between r and ϵ,

First we assume $\forall m.\ |\frac{C_m}{N} - P_m| \le r \Rightarrow P_m - r \le \frac{C_m}{N} \le P_m + r$

if for output $b_{m'}$ its $C_{m'} \ge C_{m^*}$ and to be ϵ-optimal

it's $P_{m'} \ge P_{m^*} - \epsilon$, then we know:

$$P_{m'} + r \ge P_{m^*} - r \Rightarrow P_{m'} \ge P_{m^*} - 2r \Rightarrow r = \frac{\epsilon}{2}$$

So we know:

$$2 \cdot M \cdot e^{(-2 \cdot (\frac{\epsilon}{2})^2 \cdot N)} \le \delta \Rightarrow 2 \cdot (\frac{\epsilon}{2})^2 \cdot N \ge \ln(\frac{2M}{\delta})$$

$$\Rightarrow N \ge \frac{2}{\epsilon^2} \cdot \ln(\frac{2M}{\delta})$$

13. (b)

14. (a)

15. (d)

16. (e)

17. (d)

18. (d)

19. (e)

20. (c)

```
import numpy as np
import random

filename = 'data.txt'

x_vector_array = np.ones([256, 11])
x0 = 1      #set to different number according to the problem
y_array = np.zeros(256)
with open(filename) as file:
    a = 0
    for line in file:
        line = line.strip().split()
        for i in range(10):
            x_vector_array[a][i+1] = line[i]
        x_vector_array[a][0] = x0
        y_array[a] = line[10]
        a += 1
file.close()

#x_vector_array = x_vector_array / 2    #for scaling if needed

N = 256
M = N * 4       #set to different number according to the problem
total_E = 0
update_num_array = np.zeros(1000)
w0_array = np.zeros(1000)
for i in range(1000):
    correct_times = 0
    w_vector = np.zeros(11)
    update_num = 0
    while correct_times < M:
        r_seed = random.randint(0, 255)
        sign_for_wTx = np.sign(np.dot(w_vector, x_vector_array[r_seed]))
        if (sign_for_wTx != -1 and y_array[r_seed] == 1) or (sign_for_wTx == -1
and y_array[r_seed] == -1):
            correct_times += 1
            continue
        else:
            correct_times = 0
            w_vector = w_vector + y_array[r_seed] * x_vector_array[r_seed]
            update_num += 1
    update_num_array[i] = update_num
    w0_array[i] = w_vector[0] * x0      #set to different number according to the
problem
    wrong_sign = 0
    for j in range(256):
        sign_for_wTx = np.sign(np.dot(w_vector, x_vector_array[j]))
        if (sign_for_wTx == -1 and y_array[j] == 1) or (sign_for_wTx != -1 and
y_array[j] == -1):
            wrong_sign += 1
    total_E += wrong_sign / 256
average_E = total_E / 1000
```

```
print(average_E)
print(np.median(update_num_array))
print(np.median(w0_array))
```