1. c

(a)
$$\begin{bmatrix} 7 & 8 & 9 \\ 17 & 18 & 19 \\ 27 & 28 & 29 \end{bmatrix}$$

(b)
$$\begin{bmatrix} 1 & 1 & 1 \\ 7 & 8 & 9 \\ 15 & 16 & 17 \\ 21 & 23 & 25 \end{bmatrix}$$

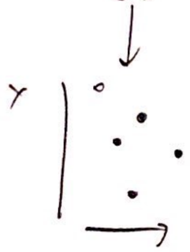
(c)
$$\begin{bmatrix} 1 & 1 & 3 \\ 7 & 8 & 9 \\ 15 & 16 & 17 \\ 21 & 23 & 25 \end{bmatrix}$$

(d)
$$\begin{bmatrix} 1 & 3 & 5 \\ 7 & 8 & 9 \\ 15 & 16 & 17 \\ 21 & 23 & 25 \end{bmatrix}$$

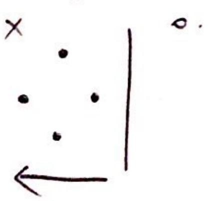
(e)
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 15 & 16 & 17 \\ 21 & 23 & 25 \end{bmatrix}$$

2. d

$$4(N-1)+2 = 4N-2$$



or



3. d.

当 $w_0 > 0$ 時. 只会使能被 shatter 的 H 減少. 因此仍有 H 可被 shatter.
所以 d_{vc} 跟一般的 2D perceptron 一樣為 $d+1 \to 3$.

4. b.

只需要指定 a b 兩個边界值
因此為 $\binom{N+1}{2} + 1$

5. b.

$d_{vc} \approx$ free parameters.
承上題. 因此 $d_{vc} = 2$.

6. d.

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

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

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

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

$$E_{in}(g) - \varepsilon \le E_{out}(g) \le E_{in}(g) + \varepsilon \quad — \text{①}$$

$$E_{in}(g^*) - \varepsilon \le E_{out}(g^*) \le E_{in}(g^*) + \varepsilon \quad — \text{②}$$

$\times \ominus$

$$-E_{in}(g^*) - \varepsilon \le -E_{out}(g^*) \le -E_{in}(g^*) + \varepsilon \quad — \text{③}$$

① + ③

$$\to E_{in}(g) - E_{in}(g^*) - 2\varepsilon \le E_{out}(g) - E_{out}(g^*) \le E_{in}(g) - E_{in}(g^*) + 2\varepsilon$$

$$E_{in}(g) - E_{in}(g^*) < 0$$

$$E_{out}(g) - E_{out}(g^*) \le 2\varepsilon$$

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

7. d.

∵ binary classifier

$d_{vc} = 0 \to M = 2^0 = 1$ $\qquad d_{vc} = \log_2 M$

$d_{vc} = 1 \to M = 2^1 = 2$

$d_{vc} = 2 \to M = 2^2 = 4$

$d_{vc} = 3 \to M = 2^3 = 8$

$\vdots$

8. d.

$\{-1, +1\}^k \to k$ dimension.

$d_{vc} = k+1$

9. b.

第一條: some set of d distinct inputs is shattered by H.

→ 以 N=3 的 2D perceptron 為例. 最多有 8 种, 但∵共線時, 只有 6 种.

最後一條: any set of d+1 distinct inputs is not shattered by H.

→ ∵ $d_{vc}(H) = d$ ∴ d+1 絕不能 shattered

10. C

sin 函数透过不同的運算. 能夠 shattered 任意點.

11. d.

$$\bar{E}(h,0)(1-\tau)+(1-E(h,0))\tau = E_{out}(h,\tau)$$

$$(1-\tau)E(h,0)+\tau-\tau E(h,0) = E_{out}(h,\tau)$$

$$E(h,0)(1-2\tau) = E_{out}(h,\tau)-\tau$$

$$E(h,0) = \frac{E_{out}(h,\tau)-\tau}{1-2\tau}$$

12. b.

$f(x)=1.$

$(2-1)^2\cdot 0.1+(3-1)^2\cdot 0.2 = 0.9$

$f(x)=2.$

$(3-2)^2\cdot 0.1+(1-2)^2\cdot 0.2 = 0.3$

$f(x)=3$

$(1-3)^2\cdot 0.1+(2-3)^2\cdot 0.2 = 0.6$

$(0.9+0.3+0.6)\cdot\frac{1}{3}$

$= 0.6$

13. b.

$f(x)=1$

$[1-(1\cdot 0.7+2\cdot 0.1+3\cdot 0.2)]^2 = 0.25$

$f(x)=2$

$[2-(2\cdot 0.7+3\cdot 0.1+1\cdot 0.2)]^2 = 0.01$

$f(x)=3$

$[3-(3\cdot 0.7+1\cdot 0.1+2\cdot 0.2)]^2 = 0.16$

$(0.25+0.01+0.16)\cdot\frac{1}{3}$

$= 0.14$

14. d.

$$\delta = 4 \cdot (2 \cdot 2N) \cdot e^{-\frac{1}{8}\varepsilon^2 N} \rightarrow \text{run program.}$$

$N = 6000$ $\delta = 53.0961$

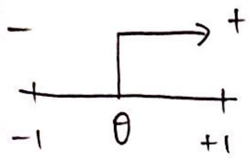
$N = 8000$ $\delta = 5.8112$

$N = 10000$ $\delta = 0.5962$

$N = 12000$ $\delta = 0.0587$

$N = 14000$ $\delta = 0.0056$

15. b.



$$E_{out}(h_{+1}, \theta, 0) = \frac{|\theta|}{1-(-1)} = \frac{|\theta|}{2}$$ (θ可正可負)

```
Q16-20
Ans: d、b、e、c、a

import random
import numpy as np
from tqdm import tqdm


def sign(number):
    if number > 0:
        return 1
    else:
        return -1


def get_data(N, probability):
    x = np.random.uniform(-1, 1, (N, 1))
    y = np.zeros((N, 1))
    arr = np.random.permutation([i for i in range(N)])

    for i in range(N):
        if i < N*probability:
            y[arr[i]][0] = -sign(x[arr[i]][0])
        else:
            y[arr[i]][0] = sign(x[arr[i]][0])
    data = np.concatenate((x, y), axis=1)
    data = np.sort(data, axis=0)

    return data


def decision_stump(data):
    N = len(data)
    min_errors = N
    min_s = 0
    min_theta = 0

    for s in [-1, 1]:
        for i in range(N):
```

```
            if i >= 0 and i <= N-2:
                if data[i][0] != data[i+1][0]:
                    theta = (data[i][0] + data[i+1][0])
 / 2
                else:
                    theta = -1
            else:
                theta = -1

            errors = 0
            for j in range(N):
                pred = s*sign(data[j][0] - theta)
                if pred != data[j][1]:
                    errors += 1
            if errors < min_errors:
                min_errors = errors
                min_s = s
                min_theta = theta
            if errors == min_errors:
                if s + theta < min_s + min_theta:
                    min_s = s
                    min_theta = theta

    error_rate = min_errors / N

    return error_rate, min_s, min_theta


def CalEout(data, s, theta):
    N = len(data)
    errors = 0
    for i in range(N):
        pred = s*sign(data[i][0] - theta)
        if pred != data[i][1]:
            errors += 1

    error_rate = errors / N

    return error_rate
```

```python
def train(N, probability):
    E_list = []
    for i in tqdm(range(100), ncols=50):

        # Ein
        N = 20
        data = get_data(N, probability)
        E_in, s, theta = decision_stump(data)

        # Eout
        N = 100000
        data = get_data(N, probability)
        E_out = CalEout(data, s, theta)

        E_diff = E_out - E_in
        E_list.append(E_diff)

    E_mean = np.mean(E_list)

    print(E_mean)


train(2, 0)
train(20, 0)
train(2, 0.1)
train(20, 0.1)
train(200, 0.1)
```