HTML HW2

1. A hypothesis fixed at point can rotate freely at any angle. Now, to separate the points, we must place the line between the ones with output 1 and the ones with output -1 . There are $N+1$ possibilities, but the output 1 and -1 is interchangable, so the answer is $2N+2$, [c].

2. For a d-dimensional perceptron, the VC dimension is $d_{VC} = d+1$, and with x_0 fixed, $d_{VC} = 6211$. But there is only 1126 hypotheses, so the tighter upper bound is $\log_2(1126)$, [a].

3. [a] For the union of two positive interval, the growth function is of order

$$m_{\mathcal{H}}(N) \sim \binom{N+1}{4} = O(N^4)$$

but according to the theorem

$$m_{\mathcal{H}}(N) = O(N^{d_{VC}})$$

we conclude that $d_{VC} = 4$. For $N = 5$, the dichotomy OXOXO can not be separated.
[b] For d-dimensional perceptrons, $d_{VC} = d+1 = 4$.
[c] The function $x - \sin(\omega x)$ oscillates between positive and negative more frequently with larger ω . As $\omega \to \infty$, the function changes sign approximately every π/ω, or infinitely often, which means with a suitable choice of ω, and the inputs constrained not far from the origin, we can find some N inputs that can be shattered by $\mathcal{H}$, which means $d_{VC} \to \infty$.
[d] For $N = 1, 2$, it can be easily shown that the inputs can be shattered. For $N = 3$, position the inputs such that the points form a right triangle with the right angle in the top left corner. This way we can shatter all inputs. For $N = 4$, position the points as a diamond.
O
O O
O
then it can be shown that the inputs can be shattered. For $N = 5$, place a negative point in the interior of the other four positive points, then it is not possible to shatter the inputs, so $d_{VC} = 4$.
[e] For $N = 3$, position the data points as follows
O O
O
then the inputs can be shattered. For $N = 4$, we can not shatter the inputs no matter how we place the points, so $d_{VC} = 3$.

The answer is [e].
4.

Sort each input by its value of $\mathbf{x}^T\mathbf{x}$, and have its label alternate between $+1$ and -1 for $2m+1$ inputs. Then the points can not be shattered, which means $d_{VC} \leq 2m$. To prove $d_{VC} \geq 2m$, we must prove that we can shatter some $2m$ inputs. We know that the value of the hypothesis alternates as $-+-+\cdots+-$ for $2m+1$ times. We again choose inputs that alternate signs every time, because consecutive $+$ or $-$ can be grouped into the same interval and the input can be regarded as alternating every time but with a shorter length. So, the choices are $-+-+\cdots-+$ or $+-+-\cdots+-$ with length $2m$, both of which can be shattered by the hypothesis $-+-+\cdots+-$, and inputs with consecutive signs is equiavlent to $-+-+\cdots$ with length $< 2m$ which can also be shattered. so we have proved $d_{VC} \leq 2m$.

The answer is [b].

5. Having a VC dimension of d means that we can not find any set of $N > d$ inputs that can be shattered. But we can find some d inputs we can shatter. So $d_{VC} \leq d$ implies that any set of $d+1$ distinct inputs is not shattered. which also implies that some set of $d+1$ distinct inputs is not shattered. Condition 6 and 8 is correct, the answer is [b].

6. We would like to minimize $E_{\text{in}}(w)$ by taking its derivative

$$\frac{dE_{\text{in}}(w)}{dw} = \frac{2}{N}\sum_{n=1}^{N} x_n(wx_n - y_n) = 0$$

   It follows that

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

   The answer is [b].

7. We would like to choose the probability distribution that is most likely to produce the given data set, which has a probability of

$$P = P(x_1) \times P(x_2) \times \cdots \times P(x_N)$$

   We vary the parameter of the distribution to find the argument for maximum P.
   [a] For a Poisson distribution,

$$P = \frac{e^{-N\lambda}\lambda^{x_1+x_2+\cdots+x_N}}{x_1!x_2!\cdots x_N!}$$

$$\frac{dP}{d\lambda} = \frac{1}{x_1!x_2!\cdots x_N!}(-Ne^{-\lambda}\lambda^{\sum_n x_n} + e^{-\lambda}\lambda^{-1+\sum_n x_n}\sum_{n=1}^{N} x_n) = 0$$

   Reorganizing

$$\lambda = \frac{1}{N}\sum_{n=1}^{N} x_n$$

[b]

$$P = (2\pi)^{-N/2} e^{-\frac{1}{2}\sum_n (x_n-\mu)^2}$$

$$\ln(P) = -\frac{N}{2}\ln(2\pi) - \frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^2$$

$$\frac{d\ln P}{d\mu} = \sum_{n=1}^{N}(x_n-\mu) = 0$$

It follows that

$$\bar{x} = \mu$$

[c]

$$P = 2^{-N} e^{-\sum_n |x_n-\mu|}$$

We would like to find the value of μ that minimizes, sort

$$\sum_{n=1}^{N} |x_n-\mu| = |x_1-\mu| + |x_2-\mu| + \cdots |x_N-\mu|$$

sort x_n by ascending order, and use the inequality

$$|a-b| + |c-b| \geq |a-c|$$

we have

$$\sum_{n=1}^{N} |x_n-\mu| \geq |x_N-x_1| + |x_{N-1}-x_2| + \cdots$$

The condition for the equal sign to hold is that μ must be the median, which is not always equal to the mean $\bar{x}$. Therefore this claim is incorrect.
[d]

$$P = (1-\theta)^{x_1+x_2+\cdots x_N - N}\theta^N$$

$$\frac{d\ln(P)}{d\theta} = \frac{d}{d\theta}\big((x_1+x_2+\cdots x_N - N)\ln(1-\theta) + N\ln\theta\big) = \frac{N}{\theta} - \frac{-N+\sum_n x_n}{1-\theta} = 0$$

solving for θ

$$\frac{1}{\theta} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

The answer is [c].

8. since $\tilde{h}(x) = \tilde{h}(-x)$,

$$P(\mathbf{x}_1)h(\mathbf{x}_1) \times P(\mathbf{x}_2)h(-\mathbf{x}_2) \times \cdots \times P(\mathbf{x}_N)h(-\mathbf{x}_N) \propto \prod_{n=1}^{N} h(y_n\mathbf{x}_n)$$

$$\max_{\mathbf{w}} \ \ln \prod_{n=1}^{N} h(y_n\mathbf{x}_n) \quad \Rightarrow \quad \min_{\mathbf{w}} \sum_{i=1}^{N} -\ln h(y_n\mathbf{x}_n)$$

Therefore

$$\tilde{E}_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(\frac{2+2|y_n\mathbf{w}^T\mathbf{x}_n|}{1+y_n\mathbf{w}^T\mathbf{x}_n+|y_n\mathbf{w}^T\mathbf{x}_n|}\right)$$

Taking its gradient

$$\frac{\partial\tilde{E}_{\text{in}}(\mathbf{w})}{\partial w_i} = -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n x_{ni}}{(|y_n\mathbf{w}^T\mathbf{x}_n|+1)(|y_n\mathbf{w}^T\mathbf{x}_n|+y_n\mathbf{w}^T\mathbf{x}_n+1)}$$

thus

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

The answer is [a].

9.

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}||\mathbf{X}\mathbf{w}-\mathbf{y}||^2$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{2}{N}(\mathbf{X}^T\mathbf{X}\mathbf{w}-\mathbf{X}^T\mathbf{y})$$

The definition of the Hessian matrix is

$$(\nabla^2 E_{\text{in}}(\mathbf{w}))_{ij} = \frac{\partial^2 E_{\text{in}}(\mathbf{w})}{\partial w_i \partial w_j}$$

let $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ and $\mathbf{b} = \mathbf{X}^T\mathbf{y}$, since

$$\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_i} = \frac{2}{N}\left(\sum_{j=1}^{N}\mathbf{A}_{ij}\mathbf{w}_j - b_i\right)$$

then

$$(\nabla^2 E_{\text{in}}(\mathbf{w}))_{ij} = \frac{\partial^2 E_{\text{in}}(\mathbf{w})}{\partial w_i \partial w_j} = \frac{2}{N}A_{ij}$$

thus

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

The answer is [b].

10.

$$(\nabla^2 E_{\text{in}}(\mathbf{w}_t))^{-1} = \frac{N}{2}\mathbf{X}^{-1}(\mathbf{X}^T)^{-1}$$

$$\mathbf{w}_1 = \mathbf{u} = -\frac{N}{2}\mathbf{X}^{-1}(\mathbf{X}^T)^{-1}\big(-\frac{2}{N}\mathbf{X}^T\mathbf{y}\big) = \mathbf{X}^{-1}\mathbf{y}$$

If we check the gradient

$$\nabla E_{\text{in}}(\mathbf{w}_1) = \frac{2}{N}(\mathbf{X}^T\mathbf{X}(\mathbf{X}^{-1}\mathbf{Y}) - \mathbf{X}^T\mathbf{y}) = \mathbf{0}$$

The answer is [a].

11. The VC bound is given by

$$P_{\mathcal{D}}\big[|E_{\text{in}} - E_{\text{out}}| > \epsilon\big] \le 4(2N)^{d_{VC}}\exp(-\epsilon^2 N/8)$$

For the decision stump model

$$\delta = 16N^2\exp(-\epsilon^2 N/8)$$

Calculating δ for each option
[a] 1.55×10^5
[b] 1.17×10^7
[c] 7.03×10^7
[d] 4.29×10^{-3}
[e] 3.07×10^{-123}

The answer is [d].

12. There are multiple scenarios depending on the range of θ and x :

- $\theta > 0.5$: Here $h(x) = -1$, therefore

$$E_{\text{out}} = P\,[\,y = +1\,] = \tau P(x < 0) + (1-\tau)P(x > 0) = \frac{1}{2}$$

- $0 < \theta < 0.5$: Consider three cases,
  1. $\theta < x \le 0.5$: Here $h(x) = +1$, thus

$$E_{\text{out}} = P\,[\,y = -1\,] = \tau$$

  2. $0 < x \le \theta : h(x) = -1$,

$$E_{\text{out}} = P\,[\,y = +1\,] = 1 - \tau$$

  3. $x < 0 : h(x) = -1$,

$$E_{\text{out}} = P[y = +1] = \tau$$

Hence,

$$E_{\text{out}} = \left(\frac{1}{2} - \theta\right)\tau + \theta(1-\tau) + \frac{1}{2}\tau = \theta(1-2\tau) + \tau$$

- $-0.5 < \theta < 0$: by symmetry,

$$E_{\text{out}} = -\theta(1-2\tau) + \tau$$

- $\theta < -0.5$: again using symmetry,

$$E_{\text{out}} = \frac{1}{2}$$

For all cases,

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

The answer is [d].

13.

```python
import numpy as np


def Ein(s, x, y, theta):
    if theta == float('-inf'):
        return np.count_nonzero(y == -s) / len(y)
    h = np.sign(s * (x - theta))
    return np.sum(h != y) / len(y)


def Eout(s, theta):
    return min(abs(theta), 0.5) if s == 1 else 1 - min(abs(theta), 0.5)


def DecisionStump():
    x = np.sort(np.random.uniform(low=-0.5, high=0.5, size=2))
    y = np.sign(x)

    thetas = [float('-inf')] + [(x[i] + x[i + 1]) / 2 for i in range(len(x) - 1)]

    minE = float('inf')
    minS = None
    minTheta = None

    for t in thetas:
        for s in [-1, 1]:
            E = Ein(s, x, y, t)
            if E < minE:
                minE = E
```

```
                minS = s
                minTheta = t
            elif E == minE and s * t < minS * minTheta:
                minE = E
                minS = s
                minTheta = t

    return Eout(minS, minTheta) - minE


mean = sum(DecisionStump() for i in range(10000)) / 10000

print(mean) # 0.29376981584936057
```

The answer is [b].

14.

```
import numpy as np


def Ein(s, x, y, theta):
    if theta == float('-inf'):
        return np.count_nonzero(y == -s) / len(y)
    h = np.sign(s * (x - theta))
    return np.sum(h != y) / len(y)


def Eout(s, theta):
    return min(abs(theta), 0.5) if s == 1 else 1 - min(abs(theta), 0.5)


def DecisionStump():
    x = np.sort(np.random.uniform(low=-0.5, high=0.5, size=128))
    y = np.sign(x)

    thetas = [float('-inf')] + [(x[i] + x[i + 1]) / 2 for i in range(len(x) - 1)]

    minE = float('inf')
    minS = None
    minTheta = None

    for t in thetas:
        for s in [-1, 1]:
            E = Ein(s, x, y, t)
            if E < minE:
                minE = E
                minS = s
```

```
                minTheta = t
            elif E == minE and s * t < minS * minTheta:
                minE = E
                minS = s
                minTheta = t

    return Eout(minS, minTheta) - minE


mean = sum(DecisionStump() for i in range(10000)) / 10000

print(mean) # 0.003907311865304553
```

The answer is [b].

15.

```
import numpy as np

tau = 0.2


def Ein(s, x, y, theta):
    if theta == float('-inf'):
        return np.count_nonzero(y == -s) / len(y)
    h = np.sign(s * (x - theta))
    return np.sum(h != y) / len(y)


def Eout(s, theta):
    return min(abs(theta), 0.5) * (1 - 2 * tau) + tau if s == 1 else 1 -
min(abs(theta), 0.5) * (1 - 2 * tau) + tau


def DecisionStump():
    x = np.sort(np.random.uniform(low=-0.5, high=0.5, size=2))
    y = np.sign(x) * np.random.choice([1, -1], size=len(x), p=[1 - tau, tau])

    thetas = [float('-inf')] + [(x[i] + x[i + 1]) / 2 for i in range(len(x) - 1)]

    minE = float('inf')
    minS = None
    minTheta = None

    for t in thetas:
        for s in [-1, 1]:
            E = Ein(s, x, y, t)
            if E < minE:
                minE = E
```

```
                minS = s
                minTheta = t
            elif E == minE and s * t < minS * minTheta:
                minE = E
                minS = s
                minTheta = t

    return Eout(minS, minTheta) - minE


mean = sum(DecisionStump() for i in range(10000)) / 10000

print(mean) # 0.5662640455969353
```

The answer is [d].

16.

```
import numpy as np

tau = 0.2


def Ein(s, x, y, theta):
    if theta == float('-inf'):
        return np.count_nonzero(y == -s) / len(y)
    h = np.sign(s * (x - theta))
    return np.sum(h != y) / len(y)


def Eout(s, theta):
    return min(abs(theta), 0.5) * (1 - 2 * tau) + tau if s == 1 else 1 -
min(abs(theta), 0.5) * (1 - 2 * tau) + tau


def DecisionStump():
    x = np.sort(np.random.uniform(low=-0.5, high=0.5, size=128))
    y = np.sign(x) * np.random.choice([1, -1], size=len(x), p=[1 - tau, tau])

    thetas = [float('-inf')] + [(x[i] + x[i + 1]) / 2 for i in range(len(x) - 1)]

    minE = float('inf')
    minS = None
    minTheta = None

    for t in thetas:
        for s in [-1, 1]:
            E = Ein(s, x, y, t)
            if E < minE:
```

```
                minE = E
                minS = s
                minTheta = t
            elif E == minE and s * t < minS * minTheta:
                minE = E
                minS = s
                minTheta = t

    return Eout(minS, minTheta) - minE


mean = sum(DecisionStump() for i in range(10000)) / 10000

print(mean) # 0.013888179740139808
```

The answer is [b].

17.

```
import numpy as np


def Ein(s, x, y, theta):
    if theta == float('-inf'):
        return np.count_nonzero(y == -s) / len(y)
    h = np.sign(s * (x - theta))
    return np.sum(h != y) / len(y)


def DecisionStump(x, y):
    thetas = [float('-inf')] + [(x[i] + x[i + 1]) / 2 for i in range(len(x) - 1)]

    minE = float('inf')
    minS = None
    minTheta = None

    for t in thetas:
        for s in [-1, 1]:
            E = Ein(s, x, y, t)
            if E < minE:
                minE = E
                minS = s
                minTheta = t
            elif E == minE and s * t < minS * minTheta:
                minE = E
                minS = s
                minTheta = t

    return minE
```

```
data = np.loadtxt("hw2_train.dat")

x = np.transpose(data)[:-1]
y = np.transpose(data)[-1]

print(np.min([DecisionStump(xi, y) for xi in x])) # 0.026041666666666668
```

The answer is [c].

18.

```
import numpy as np


def Ein(s, x, y, theta):
    if theta == float('-inf'):
        return np.count_nonzero(y == -s) / len(y)
    h = np.sign(s * (x - theta))
    return np.sum(h != y) / len(y)


def DecisionStump(x, y):
    thetas = [float('-inf')] + [(x[i] + x[i + 1]) / 2 for i in range(len(x) - 1)]

    minE = float('inf')
    minS = None
    minTheta = None

    for t in thetas:
        for s in [-1, 1]:
            E = Ein(s, x, y, t)
            if E < minE:
                minE = E
                minS = s
                minTheta = t
            elif E == minE and s * t < minS * minTheta:
                minE = E
                minS = s
                minTheta = t

    return [minE, minS, minTheta]


data = np.loadtxt("hw2_train.dat")
test = np.loadtxt("hw2_test.dat")

x = np.transpose(data)[:-1]
```

```
y = np.transpose(data)[-1]
g = np.array([DecisionStump(xi, y) for xi in x])
k = np.argmin(g[:, 0])

Eout = Ein(g[k][1], np.transpose(test)[k], np.transpose(test)[-1], g[k][2])
print(Eout) # 0.078125
```

The answer is [e].

19.

```
import numpy as np


def Ein(s, x, y, theta):
    if theta == float('-inf'):
        return np.count_nonzero(y == -s) / len(y)
    h = np.sign(s * (x - theta))
    return np.sum(h != y) / len(y)


def DecisionStump(x, y):
    thetas = [float('-inf')] + [(x[i] + x[i + 1]) / 2 for i in range(len(x) - 1)]

    minE = float('inf')
    minS = None
    minTheta = None

    for t in thetas:
        for s in [-1, 1]:
            E = Ein(s, x, y, t)
            if E < minE:
                minE = E
                minS = s
                minTheta = t
            elif E == minE and s * t < minS * minTheta:
                minE = E
                minS = s
                minTheta = t

    return [minE, minS, minTheta]


data = np.loadtxt("hw2_train.dat")
test = np.loadtxt("hw2_test.dat")

x = np.transpose(data)[:-1]
y = np.transpose(data)[-1]
g = np.array([DecisionStump(xi, y) for xi in x])
```

```
print(np.max(g[:, 0])-np.min(g[:, 0])) # 0.30208333333333333
```

The answer is [d].

20.

```
import numpy as np


def Ein(s, x, y, theta):
    if theta == float('-inf'):
        return np.count_nonzero(y == -s) / len(y)
    h = np.sign(s * (x - theta))
    return np.sum(h != y) / len(y)


def DecisionStump(x, y):
    thetas = [float('-inf')] + [(x[i] + x[i + 1]) / 2 for i in range(len(x) - 1)]

    minE = float('inf')
    minS = None
    minTheta = None

    for t in thetas:
        for s in [-1, 1]:
            E = Ein(s, x, y, t)
            if E < minE:
                minE = E
                minS = s
                minTheta = t
            elif E == minE and s * t < minS * minTheta:
                minE = E
                minS = s
                minTheta = t

    return [minE, minS, minTheta]


data = np.loadtxt("hw2_train.dat")
test = np.loadtxt("hw2_test.dat")

x = np.transpose(data)[:-1]
y = np.transpose(data)[-1]
g = np.array([DecisionStump(xi, y) for xi in x])
b = np.argmax(g[:, 0])
k = np.argmin(g[:, 0])

Eoutb = Ein(g[b][1], np.transpose(test)[b], np.transpose(test)[-1], g[b][2])
```

```python
Eout = Ein(g[k][1], np.transpose(test)[k], np.transpose(test)[-1], g[k][2])

print(Eoutb - Eout)  # 0.34375
```

The answer is [b].