HTML HW1

1. [b] no pattern to learn from
   [c] programmable
   [d] no data
   The answer is [a].

2. The condition of the update rule is

$$\mathbf{w}_{t+1}\mathbf{x}_{n(t)}y_{n(t)} > 0$$

assume the modifier function is f, then

$$(\mathbf{w}_t + \mathbf{x}_{n(t)}y_{n(t)}f)^T\mathbf{x}_{n(t)}y_{n(t)} > 0$$

the transpose operation is linear:

$$\mathbf{w}_t^T x_{n(t)}y_{n(t)} + \mathbf{x}_{n(t)}^T x_{n(t)}y_{n(t)}^2 f > 0$$

now, $y_{n(t)}^2 = 1$, so

$$f > -\frac{\mathbf{w}_t^T x_{n(t)}y_{n(t)}}{||x_{n(t)}||^2}$$

A suitable choice is

$$f = \left\lfloor -\frac{\mathbf{w}_t^T x_{n(t)}y_{n(t)}}{||x_{n(t)}||^2} + 1 \right\rfloor$$

option [c] is incorrect because $f = -\frac{\mathbf{w}_t^T x_{n(t)}y_{n(t)}}{||x_{n(t)}||^2}$ when f is an integer.

3. ≤
   The constraint is now

$$\mathbf{w}_f^T\mathbf{w}_{t+1} = \mathbf{w}_f^T(\mathbf{w}_t + \frac{\mathbf{x}_n y_n}{||\mathbf{x}_n||})$$

$$||\mathbf{w}_{t+1}||^2 = ||\mathbf{w}_t||^2 + 2y_n\mathbf{w}_t^T\mathbf{z}_n + ||y_n(t)\mathbf{z}_n(t)||^2 \leq ||\mathbf{w}_t||^2 + 1$$

so

$$\mathbf{w}_f^T\mathbf{w}_t \geq \mathbf{w}_f^T\mathbf{w}_0 + t\rho_z||\mathbf{w}_f||$$

$$||\mathbf{w}_t||^2 \leq t$$

thus

$$1 \le \frac{\mathbf{w}_f^T \mathbf{w}_T}{||\mathbf{w}_f||\,||\mathbf{w}_T||} \le \frac{t\rho_z}{\sqrt{t}}$$

and

$$T \le \frac{1}{\rho_z^2}$$

4. $\rho = \rho_z ||\mathbf{x}_n||$, so

$$U_{\text{orig}} = \left(\frac{\max||\mathbf{x}_n||}{\rho^2}\right)^2 \ge \left(\frac{||\mathbf{x}_n||}{\rho^2}\right)^2 = U$$

The answer is [b].

5. Training examples
   $\mathbf{w}_1 = \mathbf{w}_0 + x_1 y_1 = \langle -1, 2, -2 \rangle$
   $y_2 \mathbf{w}_1^T \mathbf{x}_2 = 7$
   $y_3 \mathbf{w}_1^T \mathbf{x}_3 = 3$
   $\mathbf{w}_2 = \mathbf{w}_1 + x_3 y_3 = \langle 0, 4, -2 \rangle$
   $y_4 \mathbf{w}_2^T \mathbf{x}_4 = 4$
   $\mathbf{w}_3 = \mathbf{w}_2 + x_4 y_4 = \langle -1, 5, -2 \rangle$
   $y_5 \mathbf{w}_3^T \mathbf{x}_5 = 2$
   $\mathbf{w}_4 = \mathbf{w}_3 + x_5 y_5 = \langle 0, 6, -1 \rangle$

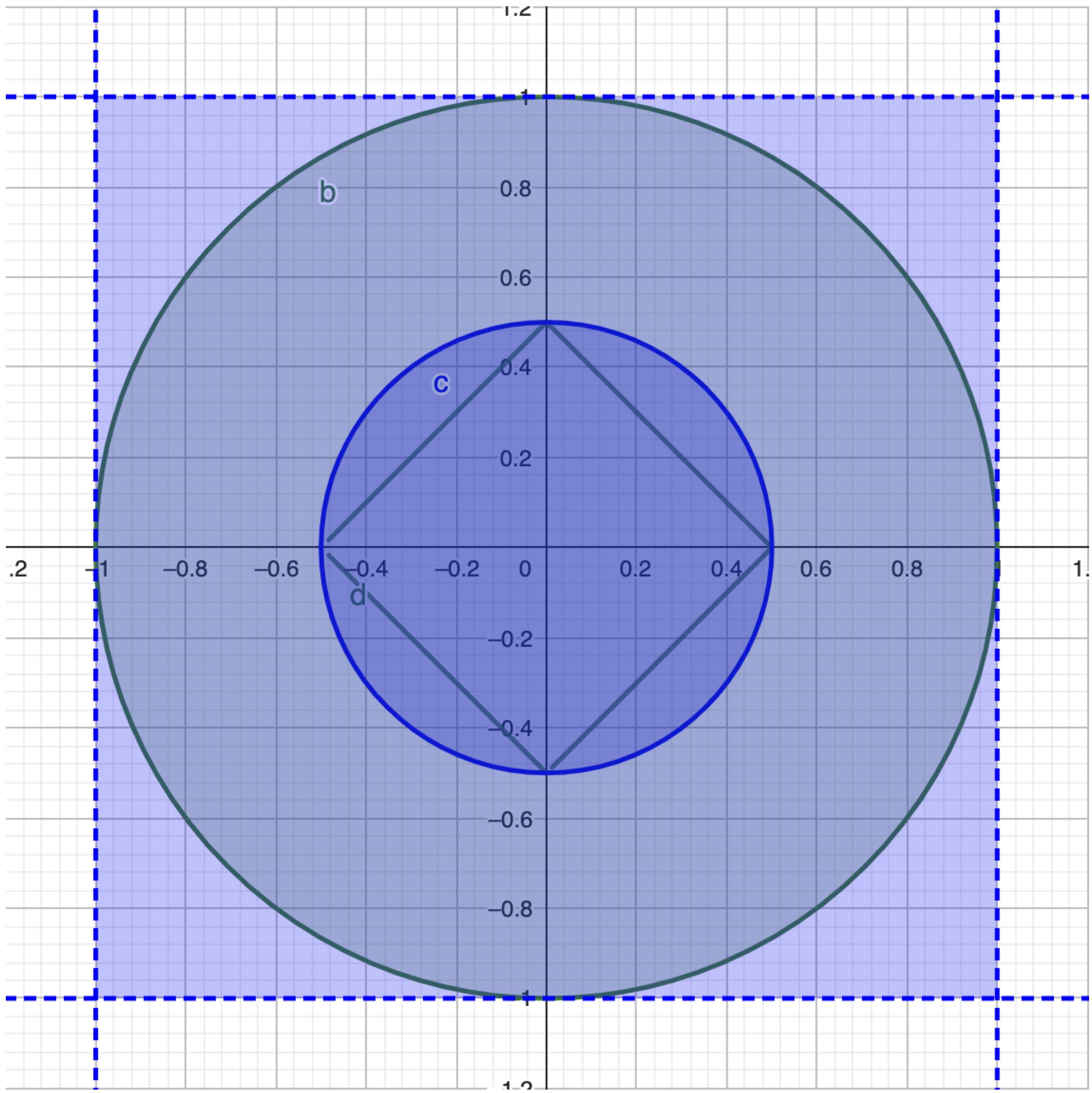
Test examples
$\mathbf{w}^T\mathbf{x}$	y
1	1
0.5	1
3	1
-4	-1

All are correctly predicted, the answer is [a].

6. There are multiple output types, and all data is labeled, so the answer is [b].

7. For the labeler, this is a case of binary classification, with the problem being "better" or "not better". Only, the classification is completed two inputs at a time. The answer is [b].

8. I select to train using the first three examples. After learning, say the line is $y - 100 = 0$ and $E_{\text{in}} = 0$ is achieved, but the other three sets of data are incorrectly labeled, $E_{\text{ots}} = 1$. Similarly, choose $y - 2.5 = 0$, then all data is correctly labeled, $E_{\text{in}} = E_{\text{ots}} = 0$. And since $0 \le E_{\text{ots}} \le 1$. We conclude that the answer is [e].

9. 

According to the figure, The error is equal to the area of the hypothesis not in the target function divided by the total area, so

$$E_{\text{out}}(h_1) = \frac{\pi(1-0.25)}{2\times 2} = \frac{3}{16}\pi$$

similarly

$$E_{\text{out}}(h_2) = \frac{\pi/4 - 0.5}{4} = \frac{\pi - 2}{16}$$

The answer is [d].

10. The probability that all data is correcly labeled is

$$\left(1-\frac{3\pi}{16}\right)^4\left(1-\frac{\pi-2}{16}\right)^4 \approx 0.021$$

The answer is [c].

11. Hoeffding's Inequality states that

$$P[|\nu-\mu|>\epsilon] \leq 2\exp(-2\epsilon^2 N)$$

The problem states that

$$P[|\nu-\pi/4|<0.01]>0.99$$

so $\epsilon=0.01$ and $2\exp(-2\epsilon^2 N)=0.01$

solving for N we get $N=26491.59$, the answer is [a].

12. Ideally for an ϵ-optimal box to be chosen, we would like the sample probability c_m/N to be close to its expected value p_m. That way boxes closer to the maximum is likely to have largest c_m. In particular, we would want boxes with $p_m < p_{m*}-\epsilon$ to deviate to the left and those with $p_m \geq p_{m*}-\epsilon$ to deviate to the write, this way we guarantee the box with largest c_m is ϵ-optimal. The probability of this happening to one box is $\exp(-2\epsilon^2 N)$, so the probability that no boxes deviate from their probability in their respective direction more than ϵ is given by

$$1-M\exp(-2\epsilon^2 N) \geq 1-\delta$$

so we have $N \geq \frac{1}{2\epsilon^2}\ln\frac{M}{\delta}$. The answer is [d].

13.

```
import numpy as np


def PLA(data):
    w = np.zeros(11)
    M = 0
    while M < 128:
        n = np.random.randint(256)
        random_data = data[n]
        x = np.insert(random_data[0:10], 0, 1)
        y = 1 if np.dot(x, w) >= 0 else -1
        if y * random_data[-1] < 0:
            M = 0
            w += x * random_data[-1]
        else:
            M += 1

    return w
```

```
def error(w):
    errors = 0
    for i in data:
        x = np.insert(i[0:10], 0, 1)
        y = 1 if np.dot(x, w) >= 0 else -1
        if y * i[-1] < 0:
            errors += 1
    return errors / 256


data = np.loadtxt("hw1_train.dat")

E_in = sum(error(PLA(data)) for _ in range(1000)) / 1000

print(E_in) #Output: 0.0198359375
```

The anwer is [b].

14.

```
import numpy as np


def PLA(data):
    w = np.zeros(11)
    M = 0
    while M < 1024:
        n = np.random.randint(256)
        random_data = data[n]
        x = np.insert(random_data[0:10], 0, 1)
        y = 1 if np.dot(x, w) >= 0 else -1
        if y * random_data[-1] < 0:
            M = 0
            w += x * random_data[-1]
        else:
            M += 1

    return w


def error(w):
    errors = 0
    for i in data:
        x = np.insert(i[0:10], 0, 1)
        y = 1 if np.dot(x, w) >= 0 else -1
        if y * i[-1] < 0:
            errors += 1
```

```python
    return errors / 256


data = np.loadtxt("hw1_train.dat")

E_in = sum(error(PLA(data)) for _ in range(1000)) / 1000

print(E_in) #Output: 0.0001953125
```

The answer is [a].

15.

```python
import numpy as np


def PLA(data):
    updates = 0
    w = np.zeros(11)
    M = 0
    while M < 1024:
        n = np.random.randint(256)
        random_data = data[n]
        x = np.insert(random_data[0:10], 0, 1)
        y = 1 if np.dot(x, w) >= 0 else -1
        if y * random_data[-1] < 0:
            M = 0
            w += x * random_data[-1]
            updates += 1
        else:
            M += 1

    return updates


data = np.loadtxt("hw1_train.dat")

print(np.median([PLA(data) for _ in range(1000)])) # Output: 449.0
```

The answer is [d].

16.

```python
import numpy as np


def PLA(data):
    w = np.zeros(11)
```

```
    M = 0
    while M < 1024:
        n = np.random.randint(256)
        random_data = data[n]
        x = np.insert(random_data[0:10], 0, 1)
        y = 1 if np.dot(x, w) >= 0 else -1
        if y * random_data[-1] < 0:
            M = 0
            w += x * random_data[-1]
        else:
            M += 1

    return w[0]


data = np.loadtxt("hw1_train.dat")

print(np.median([PLA(data) for _ in range(1000)])) #Output: 35
```

The answer is [e].

17.

```
import numpy as np


def PLA(data):
    updates = 0
    w = np.zeros(11)
    M = 0
    while M < 1024:
        n = np.random.randint(256)
        random_data = data[n]
        x = np.insert(random_data[0:10], 0, 1)/2
        y = 1 if np.dot(x, w) >= 0 else -1
        if y * random_data[-1] < 0:
            M = 0
            w += x * random_data[-1]
            updates += 1
        else:
            M += 1

    return updates


data = np.loadtxt("hw1_train.dat")

print(np.median([PLA(data) for _ in range(1000)])) # Output: 454.0
```

The answer is [d].

18.

```python
import numpy as np


def PLA(data):
    updates = 0
    w = np.zeros(11)
    M = 0
    while M < 1024:
        n = np.random.randint(256)
        random_data = data[n]
        x = np.insert(random_data[0:10], 0, 0)
        y = 1 if np.dot(x, w) >= 0 else -1
        if y * random_data[-1] < 0:
            M = 0
            w += x * random_data[-1]
            updates += 1
        else:
            M += 1

    return updates


data = np.loadtxt("hw1_train.dat")

print(np.median([PLA(data) for _ in range(1000)])) # Output: 452.0
```

The answer is [d].

19.

```python
import numpy as np


def PLA(data):
    w = np.zeros(11)
    M = 0
    while M < 1024:
        n = np.random.randint(256)
        random_data = data[n]
        x = np.insert(random_data[0:10], 0, -1)
        y = 1 if np.dot(x, w) >= 0 else -1
        if y * random_data[-1] < 0:
            M = 0
            w += x * random_data[-1]
        else:
```

```
            M += 1

    return -w[0]


data = np.loadtxt("hw1_train.dat")

print(np.median([PLA(data) for _ in range(1000)])) # Output: 34.0
```

The answer is [e].

20.

```
import numpy as np


def PLA(data):
    w = np.zeros(11)
    M = 0
    while M < 1024:
        n = np.random.randint(256)
        random_data = data[n]
        x = np.insert(random_data[0:10], 0, 0.1126)
        y = 1 if np.dot(x, w) >= 0 else -1
        if y * random_data[-1] < 0:
            M = 0
            w += x * random_data[-1]
        else:
            M += 1

    return 0.1126*w[0]


data = np.loadtxt("hw1_train.dat")

print(np.median([PLA(data) for _ in range(1000)])) # Output: 0.4310778400000001
```

The answer is [c].