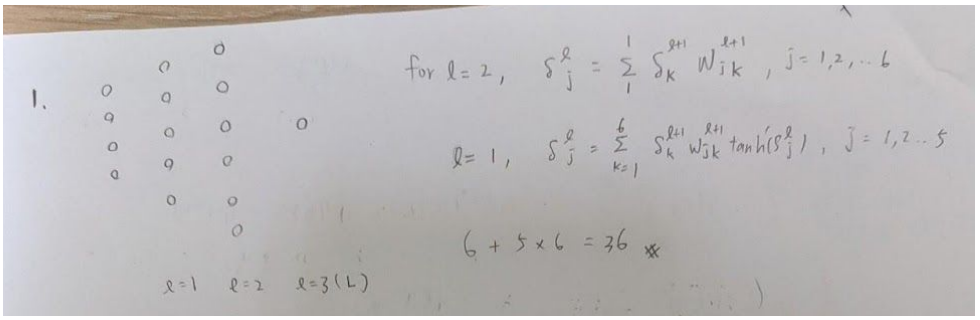
1. b, 5*6 + 6 = 36



2. e, current max = 1219, dp program
   Ans: 2 layers, first 33, second 17.

```
yiwenlai@YiWens-MacBook-Pro ~/Desktop/hw6 INSERT python3 q2.py
1219
33.0
17.0
```

   Code:

```python
import numpy as np
# dp
DP = np.zeros((51,50))
layers = np.zeros((51,50))
def max_weight(hidden, first_layer):

    out_layer = 3
    if DP[hidden][first_layer] != 0:
        return DP[hidden][first_layer]
    tmp_max = (first_layer + 1) * (hidden - 1) + hidden * out_layer
    # print((hidden,first_layer,tmp_max))
    i = 1
    current_layer = 0
    while i < hidden:
        # print((hidden-2*i-2,i+1))
        dp = max_weight(hidden - i - 1, i) + (first_layer + 1) * i
        if dp > tmp_max:
            tmp_max = dp
            current_layer = i
        DP[hidden][first_layer] = tmp_max
        layers[hidden][first_layer] = current_layer
        i += 1

    return tmp_max
if __name__ == '__main__':
    print(max_weight(50,19))

    layer = layers[50][19]
    print(layer+1)
    while layer != 1:
        if layers[int(50-int(layer)-1)][int(layer)] == 0:
            print(50-layer-1)
            break
        else:
            layer = layers[int(50 - int(layer) - 1)][int(layer)]
            print(layer)
```

3. d

3.

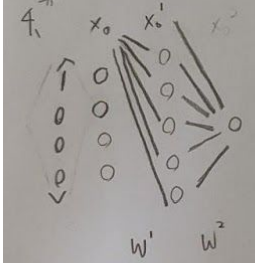
$$err(x,y) = -\sum_{k=1}^{k} v_k \ln q_k, \quad \frac{\partial err(x,y)}{\partial s_k^L} = \frac{\partial}{\partial s_k^L}\left(-v_k \ln\left(\frac{\exp s_k^L}{\Sigma \exp s_k^L}\right)\right)$$ ∵ other term will be zero due to denominator

$$\frac{d}{ds_k^L} err = \frac{\partial}{\partial s_k^L}\left[\ln(\exp s_k^L) - \ln\left(\sum_{k=1}^{k} \exp(s_k^L)\right)\right] \cdot v_k = \left(1 - \frac{\exp s_k^L}{\Sigma \exp s_k^L}\right)(v_k)$$

$$= -v_k + q_k = q_k - v_k \ ※$$

4. a, since the term is always multiplied by weight with 0, the answer is zero.

4. ※



all neurons are zeros.

$$w_{01}^1 = w_{01}^1 - \eta x_0^0 \delta_1^1 = 0 - 1 \cdot 1 \cdot \delta_1^1$$

$$\delta_1^1 = \sum_k^k \delta_k^2 w_{1k}^2 \tanh'(\ ) = 0, \quad \delta_1^2 = \delta_1^L = -2(1 - \tanh(0)) = -2(1-0) = -2$$

5. e

5.

$$\text{fix } v_n, \min_{w_m} \sum_n (r_{nm} - w_m^T v_n)^2, \quad \frac{d}{dw_m} \sum_n (r_{nm} - w_m^T v_n)^2 = 0, \quad 2\sum_n (r_{nm} - w_m^T v_n) = 0$$

$$\sum_n r_{nm} = \sum_n w_m^T v_n = \sum_n w_m^T \cdot 2 = 2n \cdot w_m^T, \quad \frac{1}{2}\frac{\sum_n r_{nm}}{n} = w_m^T \ ※$$

6. b

6. $\frac{d}{da_m} err = 2(r_{nm} - w_m^T v_n - a_m - b_n)\cdot(-1)$, $a_m \leftarrow a_m - \left(\frac{\eta}{2}\right)\frac{d}{da_m} err$

$$a_m \leftarrow a_m + \eta(r_{nm} - w_m^T v_n - a_m - b_n)$$

$$= (1-\eta)a_m + \eta(r_{nm} - w_m^T v_n - b_n) \ ※$$

7. d

7. $[0.16, 0.08, 0.24]$ $\begin{cases} g_1 g_2 & x \\ g_2 g_3 & y \\ g_1 g_3 & z \end{cases}$

$x+z<0.16$. $x+y<0.08$ $y+z<0.24$

$x+y+z=0.2$

$x=0.04$

$y=0.04$

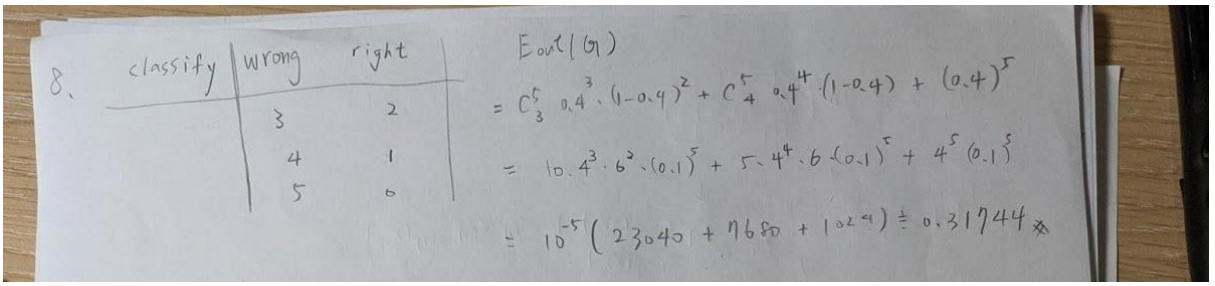
$z=0.12$ ※

x, y, z 分別是 g_1, g_2, g_3 2個分數錯誤造成的 error 數量

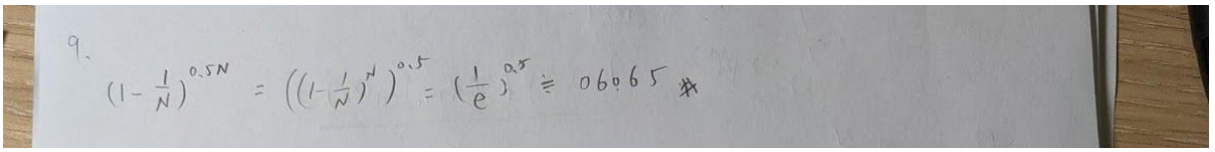
同時也是使的分類錯的情況。加上限制條件後得到此組解。

其他情形不能找到解。

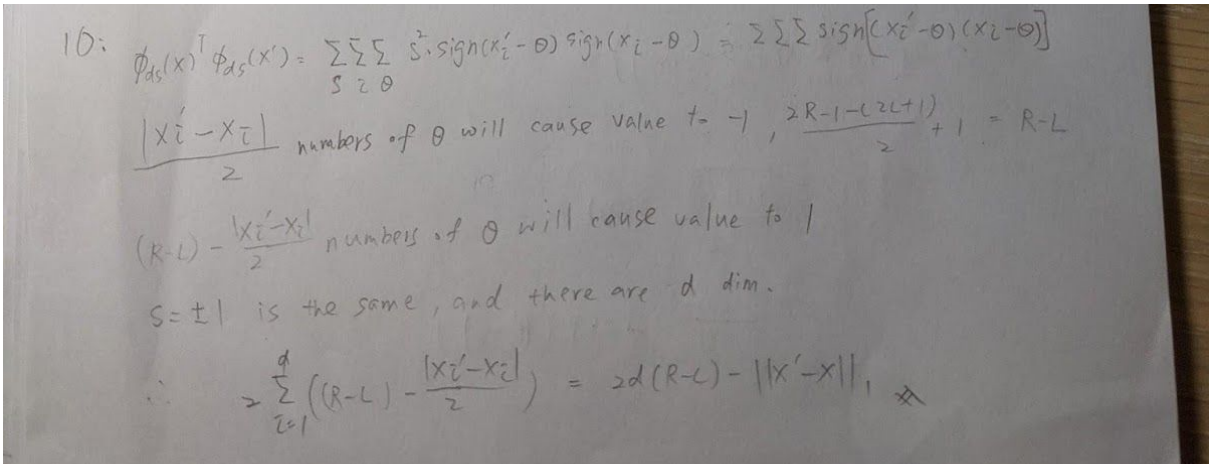
8. c



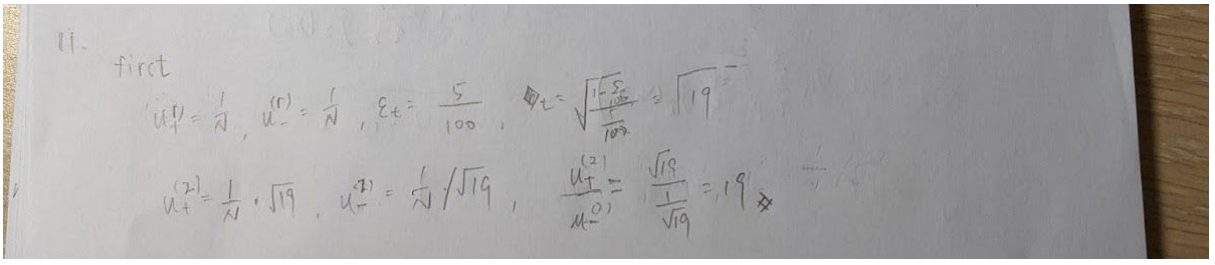
9. b



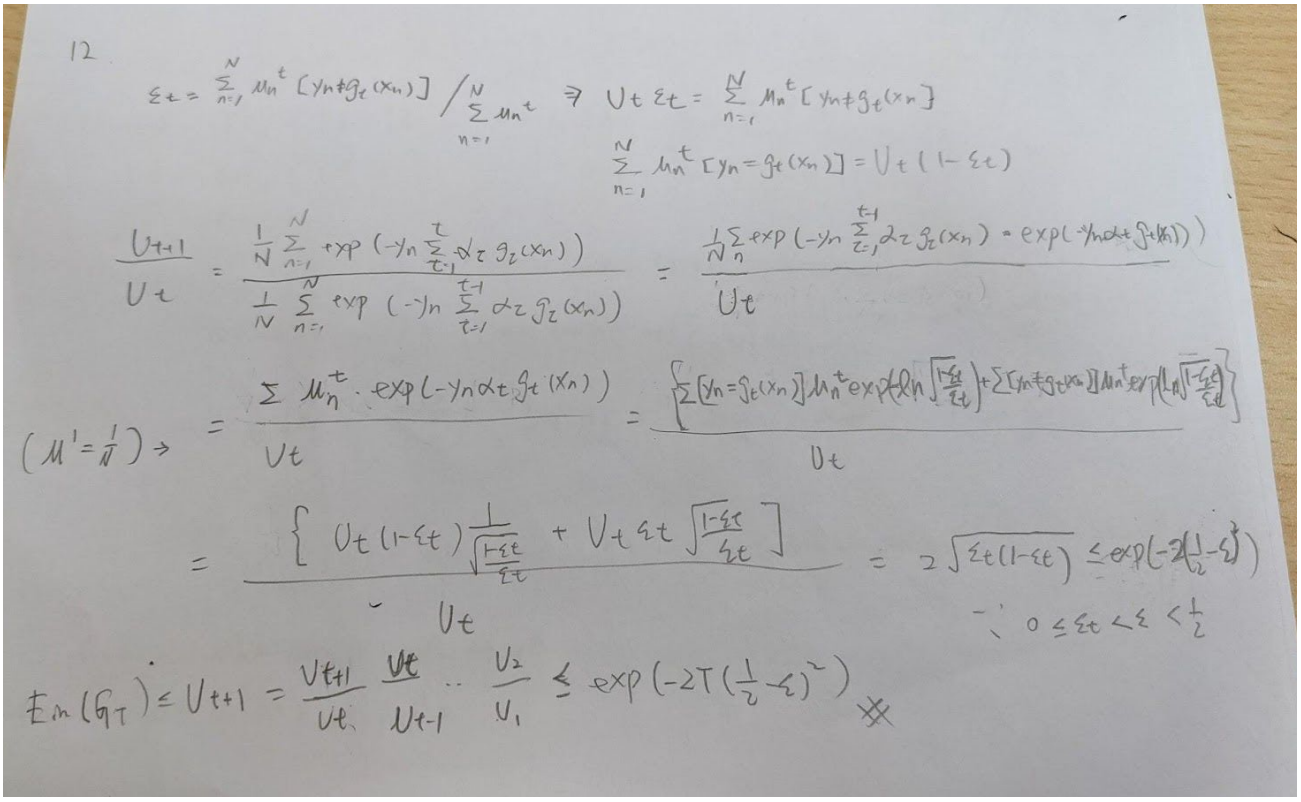
10. e, because there is no the same answer as what I derived.
The |xi' - xi| / 2 in the last row should be |xi' - xi|, since I forgot to add two cases together, so there will be no answer according to all answers except (e).



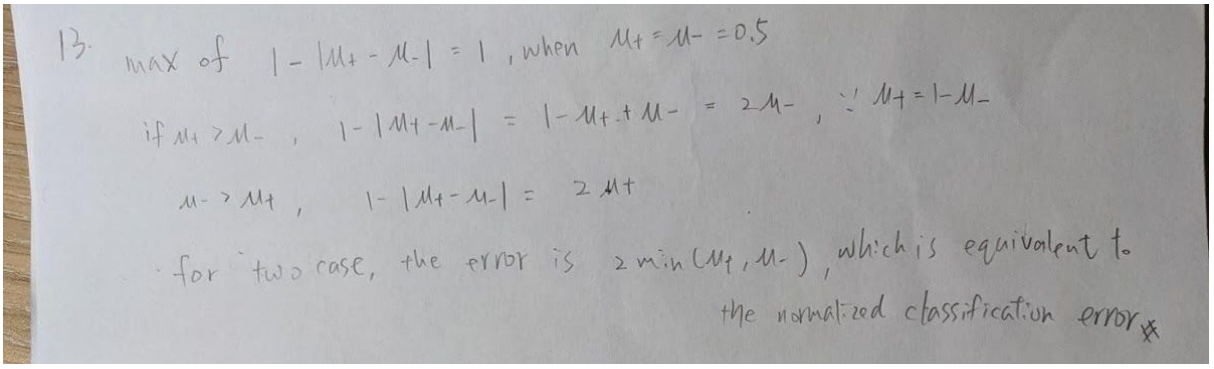
11. a



12. d



13. d



14. c

```
yiwenlai@YiWens-MacBook-Pro ~/Desktop/hw6 INSERT
14: 0.166
15: 0.229115
16: 0.014
17: 0.155
18: 0.072
```

```python
train_x, train_y, test_x, test_y = dataloader()
root = DecisionTree(train_x,train_y,'','root')
err_count = 0
for i in range(test_x.shape[0]):
    node = root
    result = 0
    while(True):
        i_feature, s, theta = node.data
        result = np.sign(test_x[i][i_feature] - theta) * s
        if result > 0:
            if node.right == None:
                # print('here')
                break
            node = node.right
        else:
            if node.left == None:
                # print('here')
                break
            node = node.left
    # print(i, result)
    if result != test_y[i]:
        err_count += 1
print('Q14: ', err_count/1000)
```

15. d

```python
# Q14 predict
print('Q14')
print('==========train==================')
root = DecisionTree(train_x,train_y,'','root')
print(root.data)
print("==========predict===============")
err_count = 0
for i in range(test_x.shape[0]):
    node = root
    result = 0
    while(True):
        i_feature, s, theta = node.data
        result = np.sign(test_x[i][i_feature] - theta) * s
        if result > 0:
            if node.right == None:
                # print('here')
                break
            node = node.right
        else:
            if node.left == None:
                # print('here')
                break
            node = node.left
    # print(i, result)
    if result != test_y[i]:
        err_count += 1
print('Eout ', err_count/1000)
# Q15
```

16. a
17. d

```
root_list = []
for t in range(10):
    bag = []
    while(len(bag)<500):
        r = np.random.randint(0, 1000, 1)
        if r not in bag:
            bag.append(r[0])
    # print(len(bag))
    part_x, part_y = [], []
    # print(train_x.shape[0])
    for i in range(train_x.shape[0]):
        if i in bag:
            part_x.append(train_x[i])
            part_y.append(train_y[i])
    part_x, part_y = np.array(part_x), np.array(part_y)
    # print(part_x.shape, part_y.shape)
    root = DecisionTree(part_x, part_y, '', 'root')
    root_list.append(root)
```

```
err_count = 0
for i in range(train_x.shape[0]):
    tmp_in = 0
    for root in root_list:
        node = root
        result = 0
        while (True):
            i_feature, s, theta = node.data
            result = np.sign(test_x[i][i_feature] - theta) * s
            if result > 0:
                if node.right == None:
                    # print('here')
                    break
                node = node.right
            else:
                if node.left == None:
                    # print('here')
                    break
                node = node.left
        # print(i, result)
        tmp_in += result
    if np.sign(tmp_in) != train_y[i]:
        err_count += 1
print('Q16: ', err_count / 1000)
```

```
err_count = 0
for i in range(test_x.shape[0]):
    tmp_out = 0
    for root in root_list:
        node = root
        result = 0
        while (True):
            i_feature, s, theta = node.data
            result = np.sign(test_x[i][i_feature] - theta) * s
            if result > 0:
                if node.right == None:
                    # print('here')
                    break
                node = node.right
            else:
                if node.left == None:
                    # print('here')
                    break
                node = node.left
        # print(i, result)
        tmp_out += result
    if np.sign(tmp_out) != test_y[i]:
        err_count += 1
print('Q17: ', err_count / 1000)
```

18. b

```
root_list = []
bag_list = []
for t in range(200):
    bag = []
    while(len(bag) < 500):
        r = np.random.randint(0, 1000, 1)
        if r not in bag:
            bag.append(r[0])
    # print(len(bag))
    part_x, part_y = [], []
    # print(train_x.shape[0])
    for i in range(train_x.shape[0]):
        if i in bag:
            part_x.append(train_x[i])
            part_y.append(train_y[i])
    part_x, part_y = np.array(part_x), np.array(part_y)
    # print(part_x.shape, part_y.shape)
    root = DecisionTree(part_x, part_y, '', 'root')
    root_list.append(root)
    bag_list.append(bag)
err_count = 0
```

```python
for i in range(train_x.shape[0]):
    # tree not trained with x[n]
    tmp_roots = []
    for j in range(len(bag_list)):
        if i not in bag_list[j]:
            tmp_roots.append(root_list[j])

    if len(tmp_roots) == 0:
        if -1 != train_y[i]:
            err_count += 1
    else:
        tmp_in = 0
        for root in tmp_roots:
            node = root
            result = 0
            while (True):
                i_feature, s, theta = node.data
                result = np.sign(test_x[i][i_feature] - theta) * s
                if result > 0:
                    if node.right == None:
                        # print('here')
                        break
                    node = node.right
                else:
                    if node.left == None:
                        # print('here')
                        break
                    node = node.left
            # print(i, result)
            tmp_in += result
        if np.sign(tmp_in) != train_y[i]:
            err_count += 1
print('Q18: ', err_count / 1000)
```

19. d

    感覺是較為成熟且許多人在使用的概念，並且有搭配上圖做說明，讓學生學習時容易理解。

20. b

    可能是課程規劃緣故，沒有做太多深入地講解，但這節的概念對初步了解推薦系統似乎有幫助，對期末比賽也是。