

1. b, $5 \times 6 + 6 = 36$

for $l=2$, $s_j^l = \sum_k s_k^{l-1} w_{jk}^{l-1}$, $j=1,2,\dots,6$

$l=3$, $s_j^l = \sum_k s_k^{l-1} w_{jk}^{l-1} \tanh(s_j^l)$, $j=1,2,\dots,5$

$6 + 5 \times 6 = 36$

$l=1$ $l=2$ $l=3(L)$

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:

```

1  import numpy as np
2  # dp
3  DP = np.zeros((51,50))
4  layers = np.zeros((51,50))
5  def max_weight(hidden, first_layer):
6
7      out_layer = 3
8      if DP[hidden][first_layer] != 0:
9          return DP[hidden][first_layer]
10     tmp_max = (first_layer + 1) * (hidden - 1) + hidden * out_layer
11     # print((hidden, first_layer, tmp_max))
12     i = 1
13     current_layer = 0
14     while i < hidden:
15         # print((hidden-2*i-2, i+1))
16         dp = max_weight(hidden - i - 1, i) + (first_layer + 1) * i
17         if dp > tmp_max:
18             tmp_max = dp
19             current_layer = i
20     DP[hidden][first_layer] = tmp_max
21     layers[hidden][first_layer] = current_layer
22     i += 1
23
24     return tmp_max
25  if __name__ == '__main__':
26     print(max_weight(50, 19))
27
28     layer = layers[50][19]
29     print(layer+1)
30     while layer != 1:
31         if layers[int(50-int(layer)-1)][int(layer)] == 0:
32             print(50-layer-1)
33             break
34         else:
35             layer = layers[int(50 - int(layer) - 1)][int(layer)]
36             print(layer)

```

3. d

3.

$$\text{err}(x, y) = - \sum_{k=1}^K y_k \ln z_k, \quad \frac{\partial \text{err}(x, y)}{\partial s_k^L} = \frac{\partial}{\partial s_k^L} \left[-y_k \ln \left(\frac{\exp s_k^L}{\sum \exp s_k^L} \right) \right]$$

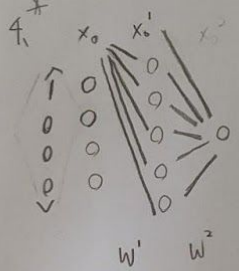
∵ other term will be zero due to denominator

$$\frac{\partial}{\partial s_k^L} \text{err} = \frac{\partial}{\partial s_k^L} \left[\ln(\exp s_k^L) - \ln \left(\sum_{k=1}^K \exp(s_k^L) \right) \right] y_k = \left(1 - \frac{\exp s_k^L}{\sum \exp s_k^L} \right) (y_k)$$

$$= -y_k + z_k = z_k - y_k$$

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

4.



all neurons are zeros.

$$w_{01} = w_{01} - \eta x_0^0 s_1^1 = 0 - 1 \cdot 1 \cdot s_1^1$$

$$s_1^1 = \sum_k s_k^2 w_{1k}^2 \tanh'(\cdot) = 0, \quad s_1^2 = s_1^L = -2(1 - \tanh(u))$$

$$= -2(1 - 0)$$

$$= -2$$

5. e

5.

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

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

6. b

6.

$$\frac{\partial}{\partial a_m} \text{err} = 2(r_{nm} - w_m^T v_n - a_m - b_n) \cdot (-1), \quad a_m \leftarrow a_m - \left(\frac{\eta}{2} \right) \frac{\partial}{\partial a_m} \text{err}$$

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

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

7. d

7.

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

$$\begin{cases} x+z < 0.16 \\ x+y < 0.08 \\ y+z < 0.24 \end{cases} \quad \begin{cases} x+y+z = 0.2 \\ x = 0.04 \\ y = 0.04 \end{cases}$$

$$z = 0.12$$

x, y, z 分別是 g_1, g_2, g_3 三個分類錯誤造成的 error 數量
同時也是使 6 個分類錯誤的情況。加上限制條件後得到此組解。
其他情形不能找到解。

8. c

8. classify

	wrong	right
3		2
4		1
5		0

$$E_{out}(G_1) = C_3^5 0.4^3 \cdot (1-0.4)^2 + C_4^5 0.4^4 \cdot (1-0.4) + (0.4)^5$$

$$= 10 \cdot 0.4^3 \cdot 0.6^2 \cdot (0.1)^5 + 5 \cdot 0.4^4 \cdot 0.6 \cdot (0.1)^5 + 4^5 (0.1)^5$$

$$= 10^{-5} (23040 + 1680 + 1024) \approx 0.31744$$

9. b

9.

$$\left(1 - \frac{1}{N}\right)^{0.5N} = \left(\left(1 - \frac{1}{N}\right)^N\right)^{0.5} = \left(\frac{1}{e}\right)^{0.5} \approx 0.6065$$

10. e, because there is no the same answer as what I derived.

The $|x_i' - x_i| / 2$ in the last row should be $|x_i' - x_i|$, since I forgot to add two cases together, so there will be no answer according to all answers except (e).

10:

$$\phi_{ds}(x)^T \phi_{ds}(x') = \sum_{S \subseteq \Theta} \sum_{i \in S} s_i \text{sign}(x_i' - \theta) \text{sign}(x_i - \theta) = \sum_{S \subseteq \Theta} \sum_{i \in S} \text{sign}[(x_i' - \theta)(x_i - \theta)]$$

$\frac{|x_i' - x_i|}{2}$ numbers of θ will cause value to -1 , $\frac{2R-1-(2L+1)}{2} + 1 = R-L$

$(R-L) - \frac{|x_i' - x_i|}{2}$ numbers of θ will cause value to 1

$S = \pm 1$ is the same, and there are d dim.

$$\therefore \sum_{i=1}^d \left((R-L) - \frac{|x_i' - x_i|}{2} \right) = 2d(R-L) - \|x' - x\|_1$$

11. a

11- first

$$u_+^{(1)} = \frac{1}{N}, u_-^{(1)} = \frac{1}{N}, \epsilon_t = \frac{5}{100}, \sigma_L = \sqrt{\frac{1 - \frac{5}{100}}{\frac{1}{100}}} = \sqrt{19}$$

$$u_+^{(2)} = \frac{1}{N} \cdot \sqrt{19}, u_-^{(2)} = \frac{1}{N} \cdot \sqrt{19}, \frac{u_+^{(2)}}{u_-^{(2)}} = \frac{\sqrt{19}}{\frac{1}{\sqrt{19}}} = 19$$

12. d

12.

$$\varepsilon_t = \frac{\sum_{n=1}^N M_n^t [y_n + g_t(x_n)]}{\sum_{n=1}^N M_n^t} \Rightarrow U_t \varepsilon_t = \frac{\sum_{n=1}^N M_n^t [y_n + g_t(x_n)]}{\sum_{n=1}^N M_n^t}$$

$$\sum_{n=1}^N M_n^t [y_n + g_t(x_n)] = U_t (1 - \varepsilon_t)$$

$$\frac{U_{t+1}}{U_t} = \frac{\frac{1}{N} \sum_{n=1}^N \exp(-y_n \sum_{\tau=1}^t \alpha_\tau g_\tau(x_n))}{\frac{1}{N} \sum_{n=1}^N \exp(-y_n \sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(x_n))} = \frac{\frac{1}{N} \sum_n \exp(-y_n \sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(x_n)) \cdot \exp(-y_n \alpha_t g_t(x_n))}{U_t}$$

$$(M' = \frac{1}{N}) \Rightarrow = \frac{\sum M_n^t \cdot \exp(-y_n \alpha_t g_t(x_n))}{U_t} = \frac{\left\{ \sum [y_n = g_t(x_n)] M_n^t \exp(-\alpha_t \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}) + \sum [y_n \neq g_t(x_n)] M_n^t \exp(-\alpha_t \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}) \right\}}{U_t}$$

$$= \frac{\left\{ U_t (1 - \varepsilon_t) \frac{1}{\sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}} + U_t \varepsilon_t \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} \right\}}{U_t} = 2 \sqrt{\varepsilon_t (1 - \varepsilon_t)} \leq \exp(-2T(\frac{1}{2} - \varepsilon)^2)$$

- ' $0 \leq \varepsilon_t < \varepsilon < \frac{1}{2}$

$$\mathbb{E}_m(\eta_T) \leq U_{t+1} = \frac{U_{t+1}}{U_t} \cdot \frac{U_t}{U_{t-1}} \cdots \frac{U_2}{U_1} \leq \exp(-2T(\frac{1}{2} - \varepsilon)^2) \quad \times$$

13. d

13. max of $1 - |M_+ - M_-| = 1$, when $M_+ = M_- = 0.5$ if $M_+ > M_-$, $1 - |M_+ - M_-| = 1 - M_+ + M_- = 2M_-$, $\because M_+ = 1 - M_-$ $M_- > M_+$, $1 - |M_+ - M_-| = 2M_+$ for two case, the error is $2 \min(M_+, M_-)$, which is equivalent to the normalized classification error \times

14. c

```

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

```

```

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

```

# 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

```

```

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。