

This homework is done by Tianwei Mo

1.

a)

$$z^H = W^H x^T + b^H = \begin{bmatrix} x_1 + x_3 \\ x_2 + x_3 \\ x_1 + x_2 - 1 \\ x_1 + x_2 + x_3 + 1 \end{bmatrix}$$

$$u_1^H = \begin{cases} 1 & \text{if } x_1 + x_3 \geq 0 \\ 0 & \text{else} \end{cases}$$

$$u_2^H = \begin{cases} 1 & \text{if } x_2 + x_3 \geq 0 \\ 0 & \text{else} \end{cases}$$

$$u_3^H = \begin{cases} 1 & \text{if } x_1 + x_2 - 1 \geq 0 \\ 0 & \text{else} \end{cases}$$

$$u_4^H = \begin{cases} 1 & \text{if } x_1 + x_2 + x_3 + 1 \geq 0 \\ 0 & \text{else} \end{cases}$$

b)

$$z^O = W^O u^{HT} + b^O = [u_1^H + u_2^H - u_3^H - u_4^H - 1.5]$$

$$\hat{y} = 1 \text{ when } z^O \geq 0,$$

that is

$$u_1^H + u_2^H > u_3^H + u_4^H + 1.5$$

$$u_1 = 1, u_2 = 1, u_3 = 0, u_4 = 0$$

Thus,

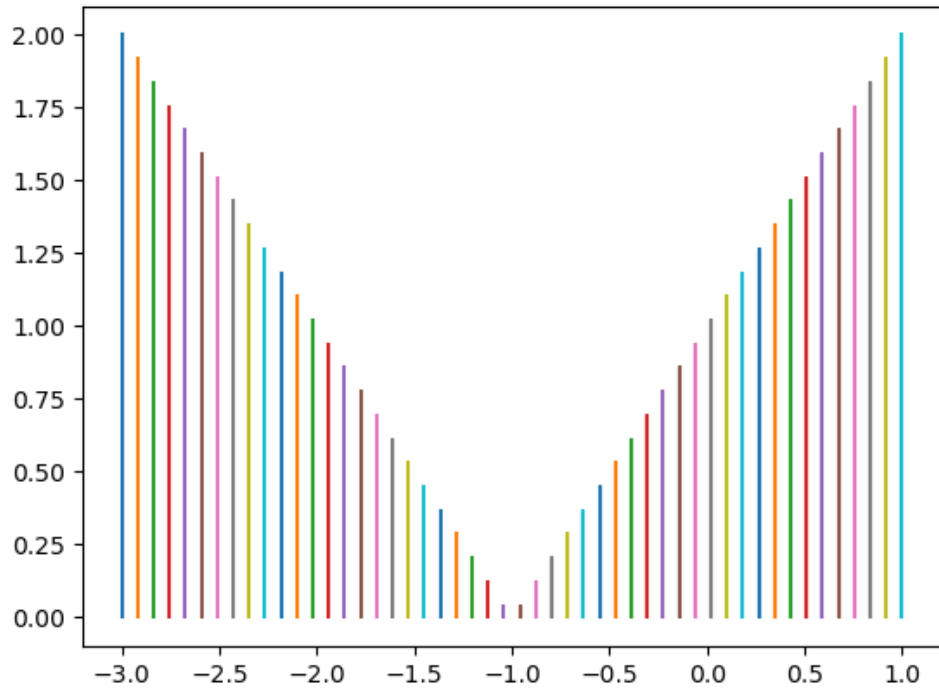
$$\begin{cases} x_1 + x_3 \geq 0 \\ x_2 + x_3 \geq 0 \\ x_1 + x_2 - 1 < 0 \\ x_1 + x_2 + x_3 + 1 < 0 \end{cases}$$

2.

a)

There are 3 hidden units.

The plot:



- b) I would choose identity function as the activation function and MSE as the loss function, that is

$$L = \frac{1}{N} \sum_{i=1}^N (y_i - z_i^o)^2$$

c)

$$x = [-2, -1, 0, 3, 3.5]$$

$$z^H = \begin{bmatrix} 1 & 0 & -1 & -4 & -4.5 \\ -1 & 0 & 1 & 4 & 4.5 \\ -4 & -3 & -2 & 1 & 1.5 \end{bmatrix}$$

$$u^H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 4.5 \\ 0 & 0 & 0 & 1 & 1.5 \end{bmatrix}$$

$$z^o = \begin{bmatrix} W_1^o & 0 & 0 & 0 & 0 \\ 0 & 0 & W_2^o & 4W_2^o & 4.5W_2^o \\ 0 & 0 & 0 & W_3^o & 1.5W_3^o \end{bmatrix} + b$$

$$z^o = [W_1^o + b \quad b \quad W_2^o + b \quad 4W_2^o + W_3^o + b \quad 4.5W_2^o + 1.5W_3^o + b]$$

$$L = \frac{1}{N} \sum_{i=1}^N (y_i - z_i^o)^2$$

$$\frac{\partial L}{\partial W^o} = -\frac{2}{N} \sum_{i=1}^N u_i^H (y_i - W_i^o u_i^H - b)$$

$$\frac{\partial L}{\partial b} = -\frac{2}{N} \sum_{i=1}^N (y_i - W_i^o u_i^H - b)$$

Let  $\frac{\partial L}{\partial W^o} = 0, \frac{\partial L}{\partial b} = 0$ :

$$W^o = [0, 1, -1], b = 0$$

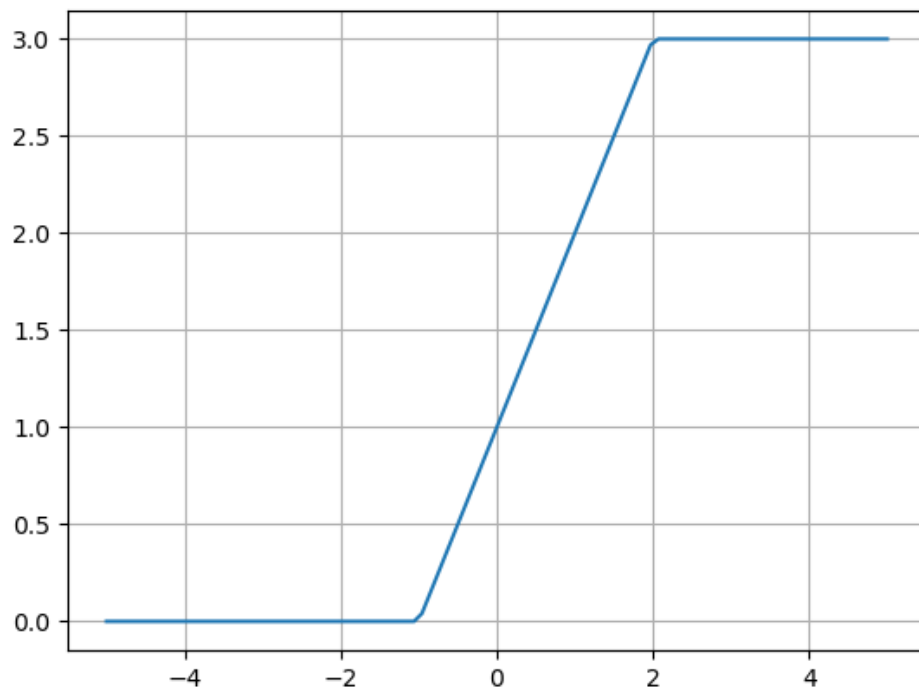
The Python code:

```

24 x = np.array([-2, -1, -0, 3, 3.5])
25 x = np.linspace(-5, 5, 100)
26 y = np.array([0, 0, 1, 3, 3])
27 wh = np.array([-1, 1, 1])
28 wh = wh[:, None]
29 bh = np.array([-1, 1, -2])
30 bh = bh[:, None]
31 wo = np.array([0, 1, -1])
32 wo = wo[:, None]
33 bo = 0
34
35 zh = wh * x + bh
36 uh = np.maximum(zh, 0)
37 zo = np.sum(wo * uh + bo, axis=0)
38 print(zo.shape)
39 yhat = zo
40
41 # c)
42 N = 5
43 dL_dw = -2 / N * np.sum(uh * (y - wo * uh - bo))
44 dL_db = -2 / N * np.sum(y - wo * uh - bo)
45

```

d) The plot:



The Python code:

```

23 x = np.array([-2, -1, -0, 3, 3.5])
24 x = np.linspace(-5, 5, 100)
25 y = np.array([0, 0, 1, 3, 3])
26 wh = np.array([-1, 1, 1])
27 wh = wh[:, None]
28 bh = np.array([-1, 1, -2])
29 bh = bh[:, None]
30 wo = np.array([0, 1, -1])
31 wo = wo[:, None]
32 bo = 0
33
34 zh = wh * x + bh
35 uh = np.maximum(zh, 0)
36 zo = np.sum(wo * uh + bo, axis=0)
37 print(zo.shape)
38 yhat = zo
39
40 plt.plot(x, yhat)
41 plt.grid()
42 plt.show()

```

e) The Python code:

```

45 def predict(x):
46     wh = np.array([-1, 1, 1])
47     wh = wh[:, None]
48     bh = np.array([-1, 1, -2])
49     bh = bh[:, None]
50     wo = np.array([0, 1, -1])
51     wo = wo[:, None]
52     bo = 0
53
54     zh = wh * x + bh
55     uh = np.maximum(zh, 0)
56     zo = np.sum(wo * uh + bo, axis=0)
57     print(zo.shape)
58     yhat = zo + zh
59
60     return yhat

```

3.

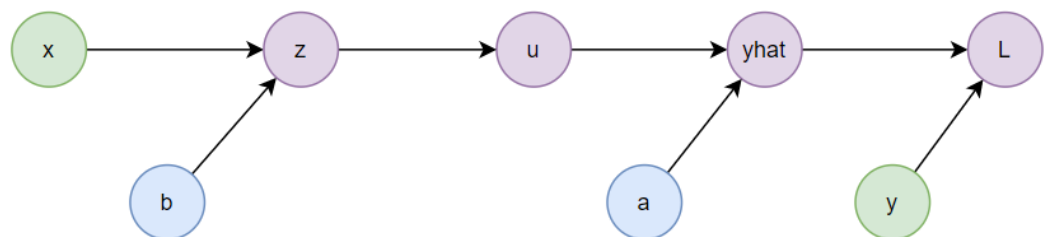
a)

$$z_{ij} = \sum_{k=1}^{N_i} w_{jk} x_{ik} + b_j$$

$$u_{ij} = \frac{1}{1 + e^{-z_{ij}}}$$

$$\hat{y}_i = \frac{\sum_{j=1}^M a_j u_{ij}}{\sum_{j=1}^M u_{ij}}$$

b) The computation graph:



Observed variable
Trainable variable
Computed variable

c)

$$\frac{\partial L}{\partial \hat{y}_i} = -2 \sum_{i=1}^N (y_i - \hat{y}_i)$$

d)

$$\frac{\partial L}{\partial u} = \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial u} = -2 \sum_{i=1}^N (y_i - \hat{y}_i) \frac{a_j \sum_{j=1}^M u_{ij} - \sum_{j=1}^M a_j u_{ij}}{\sum_{j=1}^M u_{ij}^2}$$

e)

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial u} \frac{\partial u}{\partial z} = -2 \sum_{i=1}^N (y_i - \hat{y}_i) \frac{a_j \sum_{j=1}^M u_{ij} - \sum_{j=1}^M a_j u_{ij}}{\sum_{j=1}^M u_{ij}^2} \frac{1}{(1 + e^{-z_{ij}})^2}$$

f)

$$\begin{aligned} \frac{\partial L}{\partial W_{jk}} &= \frac{\partial L}{\partial z} \frac{\partial z}{\partial W_{jk}} \\ &= -2 \sum_{i=1}^N (y_i - \hat{y}_i) \frac{a_j \sum_{j=1}^M u_{ij} - \sum_{j=1}^M a_j u_{ij}}{\sum_{j=1}^M u_{ij}^2} \frac{1}{(1 + e^{-z_{ij}})^2} \sum_{k=1}^{N_i} x_{ik} \\ \frac{\partial L}{\partial b_j} &= \frac{\partial L}{\partial z} \frac{\partial z}{\partial b_j} = -2 \sum_{i=1}^N (y_i - \hat{y}_i) \frac{a_j \sum_{j=1}^M u_{ij} - \sum_{j=1}^M a_j u_{ij}}{\sum_{j=1}^M u_{ij}^2} \frac{1}{(1 + e^{-z_{ij}})^2} N_i \end{aligned}$$

g)

$$\begin{aligned} \frac{\partial L}{\partial W_{jk}} &= \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial u} \frac{\partial u}{\partial z} \frac{\partial z}{\partial W_{jk}} \\ &= -2 \sum_{i=1}^N (y_i - \hat{y}_i) \frac{a_j \sum_{j=1}^M u_{ij} - \sum_{j=1}^M a_j u_{ij}}{\sum_{j=1}^M u_{ij}^2} \frac{1}{(1 + e^{-z_{ij}})^2} \sum_{k=1}^{N_i} x_{ik} \\ \frac{\partial L}{\partial b_j} &= \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial u} \frac{\partial u}{\partial z} \frac{\partial z}{\partial W_{jk}} \\ &= -2 \sum_{i=1}^N (y_i - \hat{y}_i) \frac{a_j \sum_{j=1}^M u_{ij} - \sum_{j=1}^M a_j u_{ij}}{\sum_{j=1}^M u_{ij}^2} \frac{1}{(1 + e^{-z_{ij}})^2} N_i \end{aligned}$$

h) The Python code:

```
16  ✓ def loss_eval(dL_dy, y, yhat, a, u):  
17      u_sum = np.sum(u)  
18      au_sum = a * u_sum  
19      a_sum_u_sum = np.sum(a * u_sum)  
20      dy_du = (a_sum_u_sum - au_sum) / u_sum ** 2  
21      dL_du = dL_dy * dy_du  
22      return dL_du
```