This homework is done by Tianwei Mo

1.

a)

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

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

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

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

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

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,

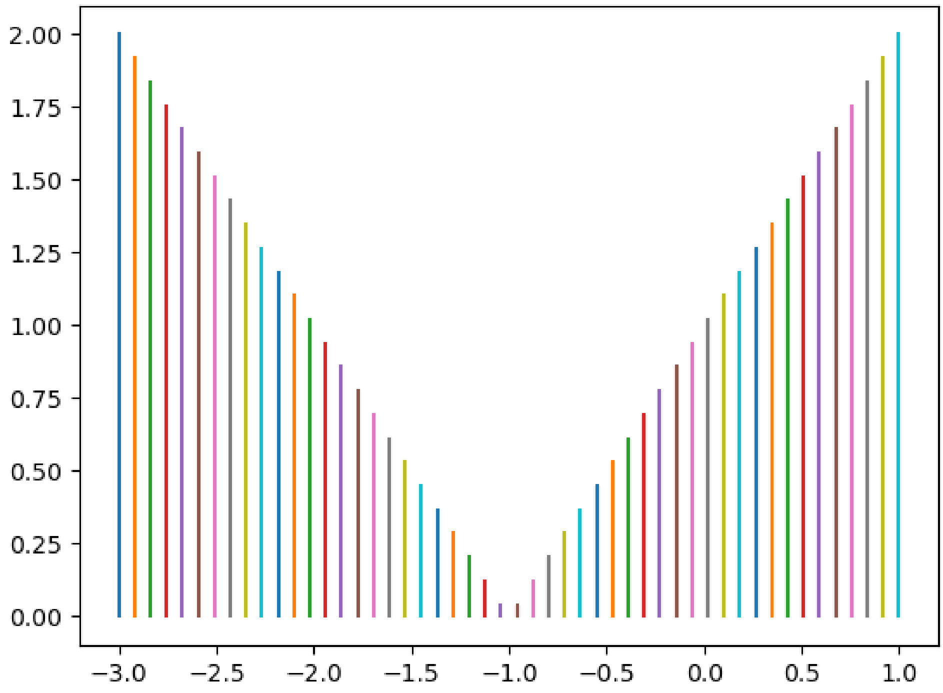
$$\begin{cases} x_1 + x_3 \geq 0 \\ x_2 + x_3 \geq 0 \\ x_1 + x_2 - 1 < 0 \\ x_1 + x_2 + x_3 + 1 < 0 \end{cases}$$

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

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

c)

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

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

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

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

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

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

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

$$\frac{\partial L}{\partial b} = -\frac{2}{N}\sum_{i=1}^{N}\left(y_i - W_i^O u_I^H - b\right)$$

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

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

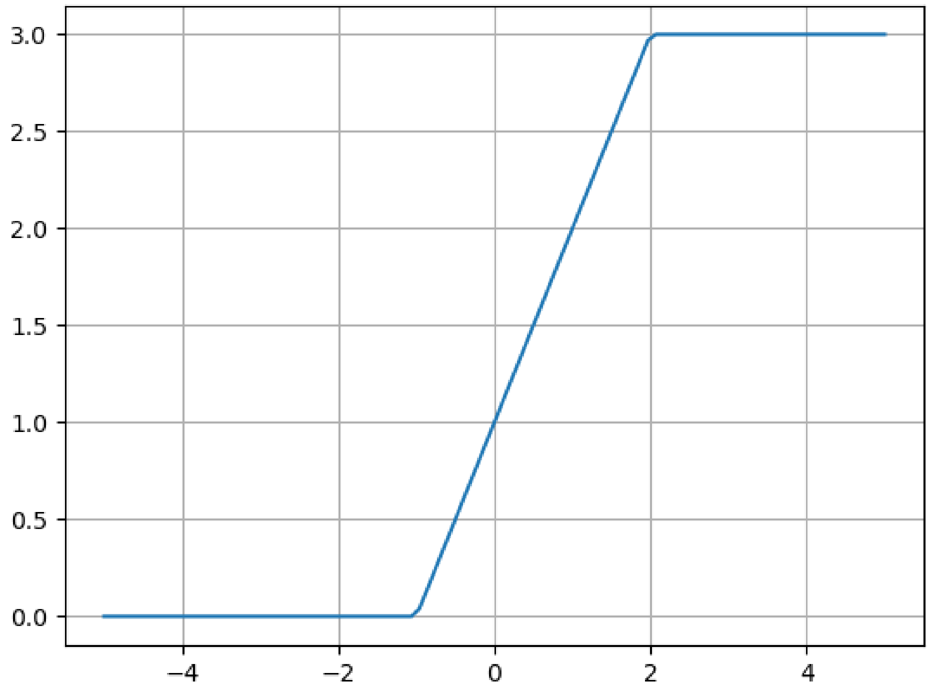
The Python code:

```
x = np.array([-2, -1, -0, 3, 3.5])
x = np.linspace(-5, 5, 100)
y = np.array([0, 0, 1, 3, 3])
wh = np.array([-1, 1, 1])
wh = wh[:, None]
bh = np.array([-1, 1, -2])
bh = bh[:, None]
wo = np.array([0, 1, -1])
wo = wo[:, None]
bo = 0

zh = wh * x + bh
uh = np.maximum(zh, 0)
zo = np.sum(wo * uh + bo, axis=0)
print(zo.shape)
yhat = zo

# c)
N = 5
dL_dw = -2 / N * np.sum(uh * (y - wo * uh - bo))
dL_db = -2 / N * np.sum(y - wo * uh - bo)
```

d) The plot:



The Python code:

```
x = np.array([-2, -1, -0, 3, 3.5])
x = np.linspace(-5, 5, 100)
y = np.array([0, 0, 1, 3, 3])
wh = np.array([-1, 1, 1])
wh = wh[:, None]
bh = np.array([-1, 1, -2])
bh = bh[:, None]
wo = np.array([0, 1, -1])
wo = wo[:, None]
bo = 0

zh = wh * x + bh
uh = np.maximum(zh, 0)
zo = np.sum(wo * uh + bo, axis=0)
print(zo.shape)
yhat = zo

plt.plot(x, yhat)
plt.grid()
plt.show()
```

e) The Python code:

```python
def predict(x):
    wh = np.array([-1, 1, 1])
    wh = wh[:, None]
    bh = np.array([-1, 1, -2])
    bh = bh[:, None]
    wo = np.array([0, 1, -1])
    wo = wo[:, None]
    bo = 0

    zh = wh * x + bh
    uh = np.maximum(zh, 0)
    zo = np.sum(wo * uh + bo, axis=0)
    print(zo.shape)
    yhat = zo + zh

    return yhat
```

3.

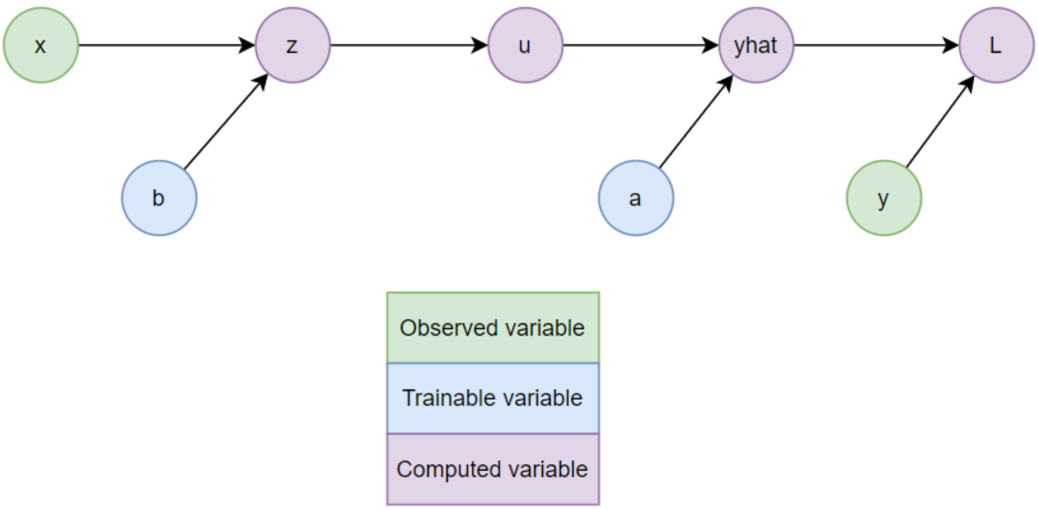
a)

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

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

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

b) The computation graph:



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)

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

f)

$$\frac{\partial L}{\partial W_{jk}} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial W_{jk}}$$

$$= -2\sum_{i=1}^{N}(y_i - \hat{y}_i)\frac{a_j \sum_{j=1}^{M} u_{ij} - \sum_{j=1}^{M} a_j u_{ij}}{\sum_{j=1}^{M} u_{ij}^{\ 2}}\frac{1}{(1 + \mathrm{e}^{-\mathrm{z_{ij}}})^2}\sum_{k=1}^{N_i} x_{ik}$$

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

g)

$$\frac{\partial L}{\partial W_{jk}} = \frac{\partial L}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial u}\frac{\partial u}{\partial z}\frac{\partial z}{\partial W_{jk}}$$

$$= -2\sum_{i=1}^{N}(y_i - \hat{y}_i)\frac{a_j \sum_{j=1}^{M} u_{ij} - \sum_{j=1}^{M} a_j u_{ij}}{\sum_{j=1}^{M} u_{ij}^{\ 2}}\frac{1}{(1 + \mathrm{e}^{-\mathrm{z_{ij}}})^2}\sum_{k=1}^{N_i} x_{ik}$$

$$\frac{\partial L}{\partial b_j} = \frac{\partial L}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial u}\frac{\partial u}{\partial z}\frac{\partial z}{\partial W_{jk}}$$

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

h) The Python code:

```python
def loss_eval(dL_dy, y, yhat, a, u):
    u_sum = np.sum(u)
    au_sum = a * u_sum
    a_sum_u_sum = np.sum(a * u_sum)
    dy_du = (a_sum_u_sum - au_sum) / u_sum ** 2
    dL_du = dL_dy * dy_du
    return dL_du
```