Hidden layer produces outputs $\mathbf{z}^{\mathrm{H}}$ and $\mathbf{u}^{\mathrm{H}}$ of dimension N_h.

Output layer produces outputs $\mathbf{z}^{\mathrm{O}}$ and $\mathbf{u}^{\mathrm{O}}$ of dimension N_o.

$$\text{Hidden layer:}\quad z_j^{\mathrm{H}} = \sum_{k=1}^{N_i} W_{jk}^{\mathrm{H}} x_k + b_j^{\mathrm{H}}, \quad u_j^{\mathrm{H}} = g_{\mathrm{act}}(z_j^{\mathrm{H}}), \quad j = 1, \ldots, N_h$$

$$\text{Output layer:}\quad z_j^{\mathrm{O}} = \sum_{k=1}^{N_h} W_{jk}^{\mathrm{O}} u_k^{\mathrm{H}} + b_j^{\mathrm{O}}, \quad u^{\mathrm{O}} = g_{\mathrm{out}}(\mathbf{z}^{\mathrm{O}}). \quad j = 1, \ldots, N_o.$$

Activation Functions:

- Hard threshold:

$$g_{\mathrm{act}}(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0. \end{cases}$$

- Sigmoid: $g_{\mathrm{act}}(z) = 1/(1+e^{-z})$.
- Rectified linear unit (ReLU): $g(z) = \max\{0, z\}$.

Output Functions:

Binary classification: In this case, the response is $y = \{0, 1\}$. To predict the response, we typically take $N_o = 1$ and the output u^{O} is the class probability:

$$P(y=1|\mathbf{x}) = u^{\mathrm{O}} = g_{\mathrm{out}}(z^{\mathrm{O}}) = \frac{1}{1+e^{-z^{\mathrm{O}}}}. \tag{3}$$

The variable z^{O} is called the *logit* and the output mapping (3) is a sigmoid. We can also use z^{O} to make a hard decision by selecting the most likely class:

$$\hat{y} = \begin{cases} 1 & z^{\mathrm{O}} \geq 0 \\ 0 & z^{\mathrm{O}} < 0. \end{cases} \tag{4}$$

K-class classification: In this case, the target is a class label $y = 1, \ldots, K$. We typically take $N_o = K$ and use the soft-max function for the class probability:

$$P(y=k|\mathbf{x}) = u_k^{\mathrm{O}} = g_{\mathrm{out,k}}(z^{\mathrm{O}}) = \frac{e^{z_k^{\mathrm{O}}}}{\sum_{\ell=1}^{K} e^{z_\ell^{\mathrm{O}}}}. \tag{5}$$

Again, we can make a hard decision on the class label by selecting the highest logit score:

$$\hat{y} = \arg\max_{k=1,\ldots,K} z_k^{\mathrm{O}}. \tag{6}$$

Regression: In this case, one is trying to predict $\mathbf{y} \in \mathbb{R}^d$, where d is the number of variables to predict and each component y_j is typically continuous-valued. For this problem, we take $N_o = d$ and $\mathbf{u}^{\mathrm{O}}$ is the prediction of $\mathbf{y}$,

$$\hat{\mathbf{y}} = \mathbf{u}^{\mathrm{O}} = g_{\mathrm{out}}(\mathbf{z}^{\mathrm{O}}) = \mathbf{z}^{\mathrm{O}}. \tag{7}$$

In (7), we will either say there is no activation or an *identity* activation.

When No = 1

$$\text{Output layer:}\quad z^{\mathrm{O}} = \sum_{k=1}^{N_h} W_k^{\mathrm{H}} u_k^{\mathrm{H}} + b^{\mathrm{O}}, \quad \hat{y} = g_{\mathrm{out}}(z^{\mathrm{O}}).$$

$$\mathbf{z}^{\mathrm{H}} = \mathbf{W}^{\mathrm{H}}\mathbf{x} + \mathbf{b}^{\mathrm{H}}, \quad \mathbf{u}^{\mathrm{H}} = g_{\mathrm{act}}(\mathbf{z}^{\mathrm{H}}),$$
$$\mathbf{z}^{\mathrm{O}} = \mathbf{W}^{\mathrm{O}}\mathbf{u}^{\mathrm{H}} + \mathbf{b}^{\mathrm{O}}, \quad \mathbf{u}^{\mathrm{O}} = g_{\mathrm{out}}(\mathbf{z}^{\mathrm{O}}).$$

$$\text{Hidden layer:}\quad z_{ij}^{\mathrm{H}} = \sum_{k=1}^{N_i} W_{jk}^{\mathrm{H}} x_{ik} + b_j^{\mathrm{H}}, \quad u_{ij}^{\mathrm{H}} = g_{\mathrm{act}}(z_{ij}^{\mathrm{H}}), \quad j = 1, \ldots, N_h$$

$$\text{Output layer:}\quad z_{ij}^{\mathrm{O}} = \sum_{k=1}^{N_h} W_{jk}^{\mathrm{O}} u_{ik}^{\mathrm{H}} + b_j^{\mathrm{O}}, \quad \mathbf{u}_j^{\mathrm{O}} = g_{\mathrm{out}}(\mathbf{z}_i^{\mathrm{O}}), \quad j = 1, \ldots, N_o.$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^{\mathsf{T}} \\ \vdots \\ \mathbf{x}_N^{\mathsf{T}} \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1,N_i} \\ \vdots & \vdots & \vdots \\ x_{N1} & \cdots & x_{N,N_i} \end{bmatrix}, \quad \mathbf{Z}^{\mathrm{H}} = \begin{bmatrix} (\mathbf{z}_1^{\mathrm{H}})^{\mathsf{T}} \\ \vdots \\ (\mathbf{z}_N^{\mathrm{H}})^{\mathsf{T}} \end{bmatrix} = \begin{bmatrix} z_{11}^{\mathrm{H}} & \cdots & z_{1,N_h}^{\mathrm{H}} \\ \vdots & \vdots & \vdots \\ z_{N1}^{\mathrm{H}} & \cdots & z_{N,N_h}^{\mathrm{H}} \end{bmatrix}$$

$$\mathrm{J}^{\mathrm{H}} = \begin{bmatrix} (\mathbf{u}_1^{\mathrm{H}})^{\mathsf{T}} \\ \vdots \\ (\mathbf{u}_N^{\mathrm{H}})^{\mathsf{T}} \end{bmatrix} = \begin{bmatrix} u_{11}^{\mathrm{H}} & \cdots & u_{1,N_h}^{\mathrm{H}} \\ \vdots & \vdots & \vdots \\ u_{N1}^{\mathrm{H}} & \cdots & u_{N,N_h}^{\mathrm{H}} \end{bmatrix}. \quad \mathbf{U}^{\mathrm{O}} = \begin{bmatrix} (\mathbf{u}_1^{\mathrm{O}})^{\mathsf{T}} \\ \vdots \\ (\mathbf{u}_N^{\mathrm{O}})^{\mathsf{T}} \end{bmatrix} = \begin{bmatrix} u_{11}^{\mathrm{O}} & \cdots & u_{1,N_o}^{\mathrm{H}} \\ \vdots & \vdots & \vdots \\ u_{N1}^{\mathrm{O}} & \cdots & u_{N,N_o}^{\mathrm{H}} \end{bmatrix}$$

$$\mathbf{Z}^{\mathrm{H}} = \mathbf{X}\,[\mathbf{W}^{\mathrm{H}}]^{\mathsf{T}} + \mathbf{B}^{\mathrm{H}}, \quad \mathbf{U}^{\mathrm{H}} = g_{\mathrm{act}}(\mathbf{Z}^{\mathrm{H}})$$
$$\mathbf{Z}^{\mathrm{O}} = \mathbf{U}^{\mathrm{H}}\,[\mathbf{W}^{\mathrm{O}}]^{\mathsf{T}} + \mathbf{B}^{\mathrm{O}} \quad \mathbf{U}^{\mathrm{O}} = g_{\mathrm{out}}(\mathbf{Z}^{\mathrm{O}})$$

$$\mathbf{B}^{\mathrm{H}} = \begin{bmatrix} (\mathbf{b}^{\mathrm{H}})^{\mathsf{T}} \\ \vdots \\ (\mathbf{b}^{\mathrm{H}})^{\mathsf{T}} \end{bmatrix} \quad \mathbf{B}^{\mathrm{O}} = \begin{bmatrix} (\mathbf{b}^{\mathrm{O}})^{\mathsf{T}} \\ \vdots \\ (\mathbf{b}^{\mathrm{O}})^{\mathsf{T}} \end{bmatrix}$$

2. Consider the data set with scalar features x_i and binary class labels $y_i = \pm 1$.

x_i	0	1.3	2.1	2.8	4.2	5.7
y_i	-1	-1	-1	1	-1	1

Consider a linear classifier for this data of the form,

$$\hat{y} = \begin{cases} 1 & z > 0 \\ -1 & z < 0, \end{cases} \qquad z = x - t,$$

where t is a threshold. For each threshold t, let $J(t)$ denote the sum hinge loss,

$$J(t) = \sum_i \epsilon_i, \quad \epsilon_i = \max(0, 1 - y_i z_i).$$

(a) Write a short python program to plot $J(t)$ vs. t for 100 values of t in the interval $t \in [0, 5]$.
(b) Based on the plot, what is one value of t that minimizes $J(t)$.
(c) For the value of t in part (b), find the corresponding slack variables ϵ_i.
(d) Which samples i violate the margin ($\epsilon_i > 0$) and which samples i are misclassified ($\epsilon_i > 1$).

2. (a) You can compute and plot $J(t)$ with the following code. The figure is shown in Fig. 1.

```
x = np.array([0,1.3,2.1,2.8,4.2,5.7])
y = np.array([-1,-1,-1,1,-1,1])
tvals = np.linspace(0,6,100)
J = []
for t in tvals:
    z = x-t
    Jt = np.sum(np.maximum(0,1-y*z))
    J.append(Jt)
J = np.array(J)

plt.plot(tvals, J)
plt.xlabel('t')
plt.ylabel('J(t)')
plt.grid()

plt.savefig('Jt.png')
```

(b) From Fig. 1, we see that $t = 4$ is a minimizer.
(c) For the value of t in part (b), find the corresponding slack variables ϵ_i.
We can compute the slack variables by the python code:

```
t = 4
z = x-t
eps = np.maximum(0, 1-y*z)
eps

>> array([ 0. ,  0. ,  0. ,  2.2,  1.2,  0. ])
```

Exponential Functions	Logarithmic Functions
$\frac{d}{dx}(e^x) = e^x$	$\frac{d}{dx}(\ln x) = \frac{1}{x}, x > 0$
$\frac{d}{dx}(a^x) = a^x \ln a$	$\frac{d}{dx}\ln(g(x)) = \frac{g'(x)}{g(x)}$
$\frac{d}{dx}(e^{g(x)}) = e^{g(x)} g'(x)$	$\frac{d}{dx}(\log_a x) = \frac{1}{x \ln a}, x > 0$
$\frac{d}{dx}(a^{g(x)}) = \ln(a)\, a^{g(x)}\, g'(x)$	$\frac{d}{dx}(\log_a g(x)) = \frac{g'(x)}{g(x)\ln a}$

2. Suppose that we are given a mini-batch of training data $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$, and we try to fit a model to the data of the form,

$$\mathbf{z}_i = \mathbf{W}\mathbf{x}_i + \mathbf{b}, \quad \hat{y}_i = \sum_{j=1}^{M} \alpha_j \exp(z_{ij}),$$

for unknown parameters $\theta = (\mathbf{W}, \mathbf{b}, \alpha)$. Here M is the dimension of the hidden variables $\mathbf{z}_i$. We use the squared error loss function,

$$L = \sum_{i=1}^{N} L_i, \quad L_i = (y_i - \hat{y}_i)^2.$$

(a) Add an intermediate variable $\mathbf{u}_i = \exp(\mathbf{z}_i)$, by which we mean $u_{ij} = \exp(z_{ij})$. Draw the computation graph showing the mapping from the inputs $\mathbf{x}_i$ and the parameters θ to the loss function, L. Also, write the equations for each step in the computation graph.

(b) Write the back-propagation equations to obtain the gradients with respect to the parameters θ.

2. (a) If we substitute $\mathbf{u}_i = \exp(\mathbf{z}_i)$ we can write the mapping from the set of $\mathbf{x}_i$'s to L as

$$\mathbf{z}_i = \mathbf{W}\mathbf{x}_i + \mathbf{b}$$
$$\mathbf{u}_i = \exp(\mathbf{z}_i), \quad \hat{y}_i = \boldsymbol{\alpha}^\mathsf{T}\mathbf{u}_i$$
$$L = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2.$$

The computation graph is shown in Fig. 2.

(b) We perform back-propagation in the following steps:

- $L \to \hat{\mathbf{y}}$: The components of the gradient are:

$$\frac{\partial L}{\partial \hat{y}_i} = 2(\hat{y}_i - y_i).$$

- $\hat{\mathbf{y}} \to \boldsymbol{\alpha}, \mathbf{u}$: We have

$$\hat{y}_i = \boldsymbol{\alpha}^\mathsf{T}\mathbf{u}_i = \sum_{j=1}^{M} \alpha_j u_{ij}.$$

Therefore,

$$\frac{\partial \hat{y}_i}{\partial u_{ij}} = \alpha_j, \quad \frac{\partial \hat{y}_i}{\partial \alpha_j} = u_{ij}.$$

Applying chain rule,

$$\frac{\partial L}{\partial \alpha_j} = \sum_{i=1}^{N} \frac{\partial L}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial \alpha_j} = \sum_{i=1}^{N} \frac{\partial L}{\partial \hat{y}_i} u_{ij}$$
$$\frac{\partial L}{\partial u_{ij}} = \frac{\partial L}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial u_{ij}} = \sum_{i=1}^{N} \frac{\partial L}{\partial \hat{y}_i}\alpha_j.$$

- $\mathbf{u} \to \mathbf{z}$: Since $u_{ij} = \exp(z_{ij})$, we have the derivative,

$$\frac{\partial u_{ij}}{\partial z_{ij}} = \exp(z_{ij}).$$

Using chain rule:

$$\frac{\partial L}{\partial z_{ij}} = \frac{\partial L}{\partial u_{ij}}\frac{\partial u_{ij}}{\partial z_{ij}} = \frac{\partial L}{\partial u_{ij}}\exp(z_{ij}).$$

- $\mathbf{z} \to \mathbf{W}, \mathbf{b}$: Since $\mathbf{z}_i = \mathbf{W}\mathbf{x}_i + \mathbf{b}$, Therefore,

$$z_{ij} = \sum_k W_{jk} x_{ik} + b_j.$$

Hence, we have the derivatives,

$$\frac{\partial z_{ij}}{\partial W_{jk}} = u_{ik}, \quad \frac{\partial z_{ij}}{\partial b_j} = 1.$$

Applying chain rule,

$$\frac{\partial L}{\partial W_{jk}} = \sum_i \frac{\partial L}{\partial z_{ij}}\frac{\partial z_{ij}}{\partial W_{jk}} = \sum_i \frac{\partial L}{\partial z_{ij}} u_{ik},$$
$$\frac{\partial L}{\partial b_j} = \sum_i \frac{\partial L}{\partial z_{ij}}\frac{\partial z_{ij}}{\partial b_j} = \sum_i \frac{\partial L}{\partial z_{ij}}.$$

2. Consider the data set for four points with scalar features x_i and binary class labels $y_i = 0, 1$.

x_i	0	1	3	5
y_i	0	0	1	0

(a) Find a neural network with $N_h = 2$ units, $N_o = 1$ output units such that $\hat{y}_i = y_i$ on all four data points. Use a hard threshold activation function (2) and threshold output function (4). State the weights and biases used in each layer.

(b) Compute the values of $\hat{y}_i$ and all the intermediate variables $\mathbf{z}_i^{\mathrm{H}}$, $\mathbf{u}_i^{\mathrm{H}}$ and z_i^{O} for each sample $x = x_i$.

(c) Now suppose we are given a new sample, $x = 3.5$. What does the network predict as $\hat{y}$?

2. (a) We can use a network similar in structure to the previous problem. In the hidden layer, we want to find features that can distinguish between $x = 3$ which belongs to class $y = 1$, and $x = \{0, 1, 5\}$ which belong to class $y = 0$. There are lots of choices, but we will extract two features: whether $x \geq 2$ and $x \geq 4$. We can do this with the linear transforms:

$$\mathbf{W}^{\mathrm{H}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad b^{\mathrm{H}} = \begin{bmatrix} -2 \\ -4 \end{bmatrix},$$

	Training				Test
x_i	0	1	3	5	3.5
$z_{i1}^{\mathrm{H}} = x - 2$	-2	-1	1	3	1.5
$z_{i2}^{\mathrm{H}} = x - 2$	-4	-3	-1	1	0.5
$u_{i1}^{\mathrm{H}} = g_{\mathrm{act}}(z_{i1}^{\mathrm{H}})$	0	0	1	1	1
$u_{i1}^{\mathrm{H}} = g_{\mathrm{act}}(z_{i2}^{\mathrm{H}})$	0	0	0	1	0
$z_i^{\mathrm{O}} = u_{i1}^{\mathrm{H}} - 2u_{i1}^{\mathrm{H}} - 0.5$	-0.5	-0.5	0.5	-1.5	0.5
$\hat{y}_i = g_{\mathrm{out}}(z_i^{\mathrm{O}})$	0	0	1	0	1
y_i	0	0	1	0	

Table 1: Computations for the neural network in Problem 2

so that

$$\mathbf{z}^{\mathrm{H}} = \mathbf{W}^{\mathrm{H}}\mathbf{x} + \mathbf{b}^{\mathrm{H}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} x + \begin{bmatrix} -2 \\ -4 \end{bmatrix} = \begin{bmatrix} x - 2 \\ x - 4 \end{bmatrix}.$$

Then, the activation outputs in the hidden layer are

$$\mathbf{u}^{\mathrm{H}} = g_{\mathrm{act}}(\mathbf{z}^{\mathrm{H}}) = \begin{bmatrix} g_{\mathrm{act}}(x-2) \\ g_{\mathrm{act}}(x-4) \end{bmatrix} = \begin{bmatrix} \mathbb{1}_{\{x \geq 2\}} \\ \mathbb{1}_{\{x \geq 4\}} \end{bmatrix}.$$

Now we need to combine these functions in a way that they are negative for samples x_i when $y_i = 0$ and positive on sample x_i when $y_i = 1$. Similar to the previous problem, take

$$W^{\mathrm{O}} = [1, -2], \quad b^{\mathrm{O}} = -0.5.$$

Then,

$$z^{\mathrm{O}} = W^{\mathrm{O}}\mathbf{u}^{\mathrm{H}} + b^{\mathrm{O}} = [1, -2]\begin{bmatrix} \mathbb{1}_{\{x \geq 2\}} \\ \mathbb{1}_{\{x \geq 4\}} \end{bmatrix} - 0.5$$
$$= \mathbb{1}_{\{x \geq 2\}} - 2\mathbb{1}_{\{x \geq 4\}} - 0.5 = \begin{cases} -0.5 & \text{when } x < 2 \\ 0.5 & \text{when } x \in [2, 4) \\ -1.5 & \text{when } x \geq 4. \end{cases}$$

The predicted output $\hat{y}$ is

$$\hat{y} = \mathbb{1}_{\{z^{\mathrm{O}} \geq 0\}} = \begin{cases} 1 & \text{if } x \in [2, 4) \\ 0 & \text{else} \end{cases}$$

This matches the training data.

(b) Table 1 computes the values for all intermediate variables for the four training data points. We can see that $\hat{y}_i = y_i$ for all four points.

(c) The value for $x = 3.5$ is shown in the final column of Table 1. For this value of $x = 3.5$, we get $\hat{y} = 1$.