

# Lab: Forward Pass

ACTL3143 & ACTL5111 Deep Learning for Actuaries

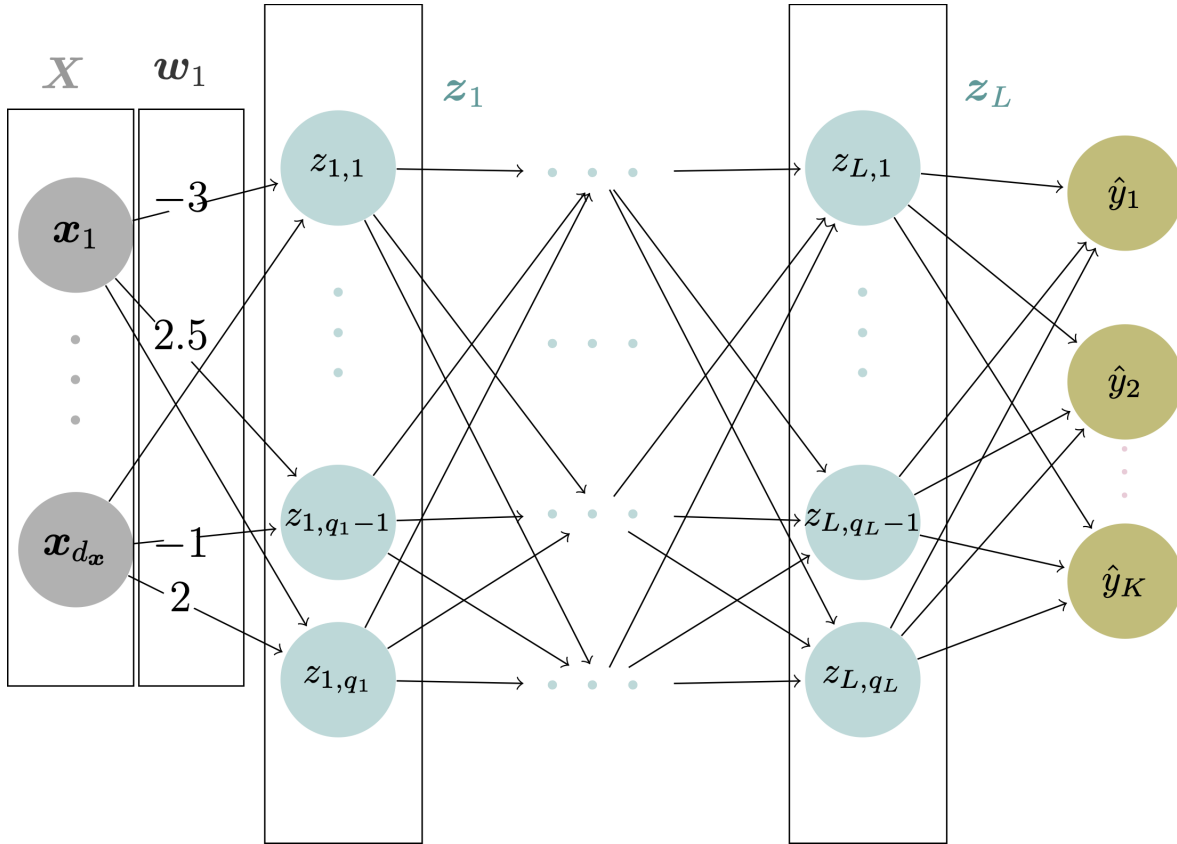


Figure 1: The structure of a neural network.

At each node in the hidden and output layers, the value  $z$  is calculated as a weighted sum of the node outputs in the previous layer, plus a bias. In other words:

$$z = Xw + b$$

where  $\mathbf{X}$  is a  $n \times d_x$  matrix representing the weights,  $\mathbf{w}$  is an  $d_x \times q$  matrix representing the weights ( $q$  representing the number of neurons in the current layer), and  $\mathbf{b}$  is an  $n \times q$  matrix representing the biases.  $n$  represents the number of observations and  $d_x$  represents the dimension of the input.

## Example: Calculate the Neuron Values in the First Hidden Layer

$$\mathbf{X} = \begin{pmatrix} 1 & 2 \\ 3 & -1 \end{pmatrix}, \mathbf{w} = \begin{pmatrix} 2 \\ -1 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

We can calculate the neuron value as  $\mathbf{z}$  follows:

$$\begin{aligned} \mathbf{z} &= \mathbf{X}\mathbf{w} + \mathbf{b} \\ &= \begin{pmatrix} 1 & 2 \\ 3 & -1 \end{pmatrix} \begin{pmatrix} 2 \\ -1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 2 \\ 3 & -1 \end{pmatrix} \begin{pmatrix} 2 \\ -1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 8 \end{pmatrix} \end{aligned}$$

Alternatively, one can use Python:

```
import numpy as np
X = np.array([[1, 2], [3, -1]])
w = np.array([[2], [-1]])
b = np.array([[1], [1]])
print(X @ w + b)
```

```
[[1]
 [8]]
```

## Exercises

1. ( $2 \times 2$  matrices) Calculate  $\mathbf{z}$ , given:

$$\begin{aligned}
1. \mathbf{X} &= \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \mathbf{w} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \mathbf{b} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\
2. \mathbf{X} &= \begin{pmatrix} 1 & -1 \\ 0 & 5 \end{pmatrix} \mathbf{w} = \begin{pmatrix} -1 \\ 8 \end{pmatrix} \mathbf{b} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}
\end{aligned}$$

2. ( $3 \times 3$  matrices) Calculate  $\mathbf{z}$ , given:

$$\begin{aligned}
1. \mathbf{X} &= \begin{pmatrix} 4 & 4 & 0 \\ 2 & 2 & 4 \\ 2 & 4 & 1 \end{pmatrix} \mathbf{w} = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} \mathbf{b} = \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix} \\
2. \mathbf{X} &= \begin{pmatrix} 6 & -6 & -2 \\ -3 & -1 & -5 \\ 1 & 1 & -7 \end{pmatrix} \mathbf{w} = \begin{pmatrix} 4 \\ 4 \\ -8 \end{pmatrix} \mathbf{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}
\end{aligned}$$

3. (non-square matrices) Calculate  $\mathbf{z}$ , given:

$$\begin{aligned}
1. \mathbf{X} &= \begin{pmatrix} 1 & 0 & 1 \\ 1 & 2 & 1 \end{pmatrix} \mathbf{w} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \mathbf{b} = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \\
2. \mathbf{X} &= \begin{pmatrix} 1 & -1 \\ 0 & 5 \\ 2 & -2 \end{pmatrix} \mathbf{w} = \begin{pmatrix} 5 \\ -7 \end{pmatrix} \mathbf{b} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}
\end{aligned}$$

4. If  $\mathbf{X}$  is a  $2 \times 3$  matrix, what does this say about the neural network's architecture? What about a  $3 \times 2$  matrix?

## Activation Functions

The result of  $\mathbf{z} = \mathbf{X}\mathbf{w} + \mathbf{b}$  will be in the range  $(-\infty, \infty)$ . However, sometimes we might want to constrain the values of  $\mathbf{z}$ . We apply an **activation function** to  $\mathbf{z}$  to do this. Activation functions include:

- Sigmoid:  $S(z_i) = \frac{1}{1+e^{-z_i}}$ , constrains each value in  $\mathbf{z}$  to  $(0, 1)$
- Tanh:  $\tanh(z_i) = \frac{e^{2z_i}-1}{e^{2z_i}+1}$ , constrains each value in  $\mathbf{z}$  to  $(-1, 1)$ .
- ReLU:  $\text{ReLU}(z_i) = \max(0, z_i)$ , only activates for a value of  $\mathbf{z}$  if it is positive.
- Softmax:  $\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$ . This maps the values in  $\mathbf{z}$  so that each value is in  $[0, 1]$  and the sum is equal to 1. This is useful for representing probabilities and is often used for the output layer.

### Example: Applying Activation Functions

Given  $\mathbf{z} = \begin{pmatrix} 1 \\ 8 \end{pmatrix}$ , calculate the resulting vector  $\mathbf{a} = \text{activation}(\mathbf{z})$  using the four activation functions above.

- Sigmoid:

$$S(\mathbf{z}) =$$

- Tanh:

$$\tanh(\mathbf{z}) =$$

- ReLU

$$\text{ReLU}(\mathbf{z}) =$$

- Softmax

$$\sigma(\mathbf{z}) =$$

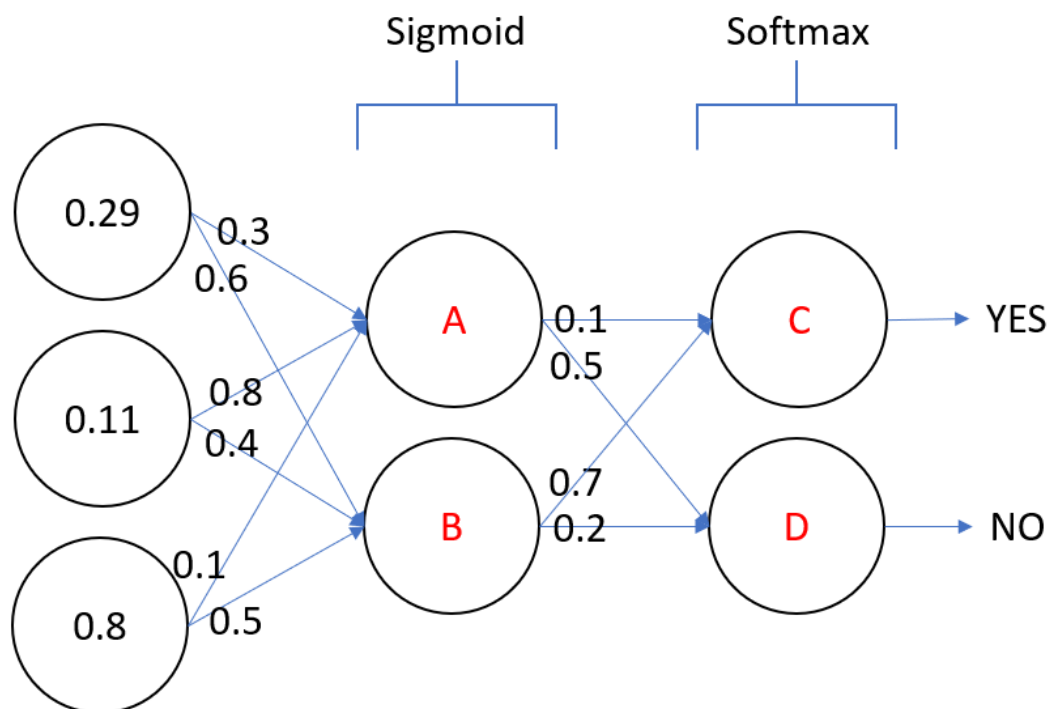
### Exercises

1. Given  $\mathbf{z} = \begin{pmatrix} 8 \\ 6 \end{pmatrix}$ , calculate the resulting vector  $\mathbf{a} = \text{activation}(\mathbf{z})$  using the four activation functions above.
2. Given  $\mathbf{z} = \begin{pmatrix} -8 \\ 9 \\ -3 \end{pmatrix}$ , calculate the resulting vector  $\mathbf{a} = \text{activation}(\mathbf{z})$  using the four activation functions above.

3. For extra practice, try calculating the vector  $\mathbf{a}$ , using the results of the exercises in section 1.

## Final Output

**Example: Calculate the Final Output**



1. With the activations, weights, and activation functions given in the above figure and a constant bias of 1 for each node, calculate the values of **A**, **B**, **C**, and **D**.
2. If the **C** node represents “YES” and the **D** node represents “NO”, what final value is predicted by the neural network?

**Hint:** Write out

1. The input matrix  $\mathbf{X}$  (should be  $1 \times 3$ ):

$$\mathbf{X} = \left( \begin{array}{ccc} \end{array} \right).$$

2. The weight matrix  $\mathbf{w}_1$  between the input layer and the first hidden layer (should be  $3 \times 2$ ):

$$\mathbf{w}_1 = \begin{pmatrix} & \\ & \\ & \end{pmatrix}, \mathbf{b}_1 = \begin{pmatrix} \\ \end{pmatrix}.$$

3. The weight matrix  $\mathbf{w}_2$  between the first hidden layer and the output layer (should be  $2 \times 2$ ):

$$\mathbf{w}_2 = \begin{pmatrix} & \\ & \end{pmatrix}, \mathbf{b}_2 = \begin{pmatrix} \\ \end{pmatrix}.$$

See more details in `Week_2_Tutorial_Notebook`