Lab: Forward Pass

ACTL3143 & ACTL5111 Deep Learning for Actuaries

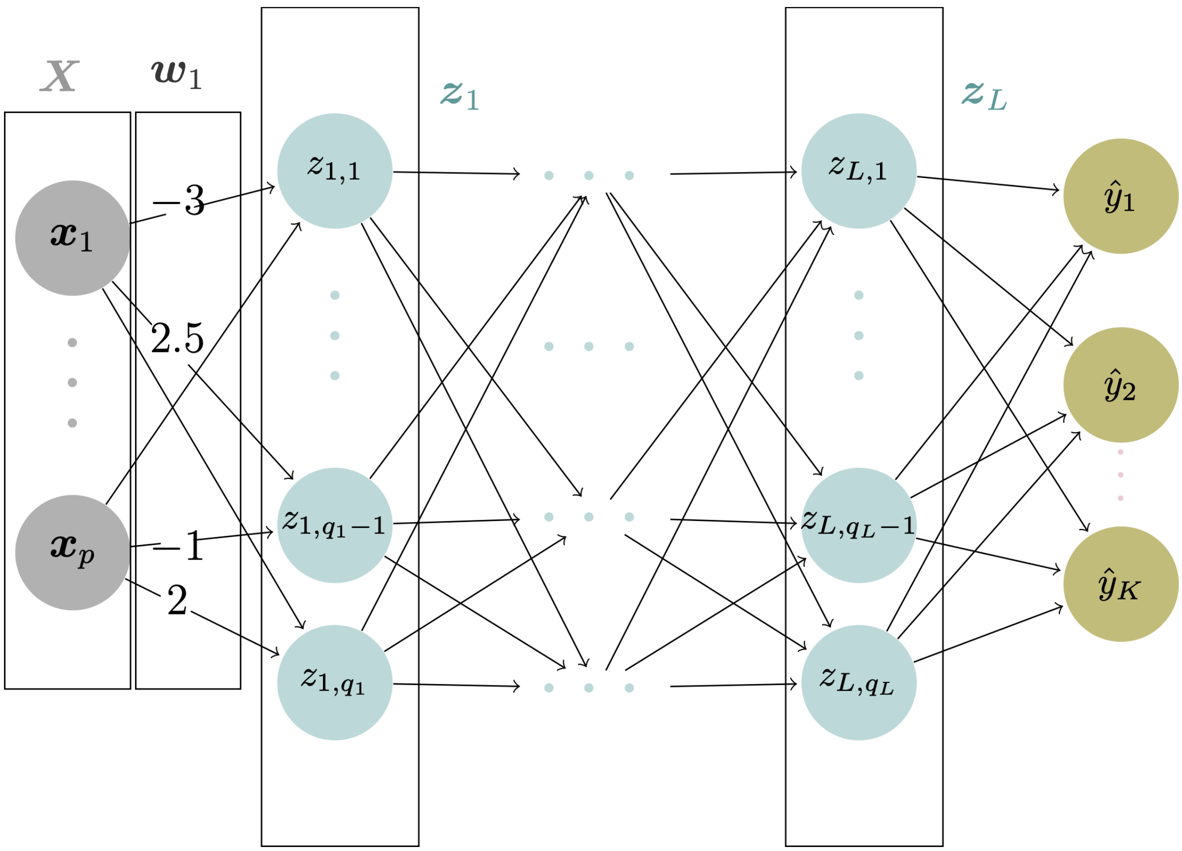


Figure 1: The structure of a neural network.

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

$$\boldsymbol{z} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{b}$$

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

Example: Calculate the Neuron Values in the First Hidden Layer

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

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

$$\begin{aligned} \boldsymbol{z} &= \boldsymbol{X}\boldsymbol{w} + \boldsymbol{b} \\ &= \begin{pmatrix} \quad & \quad & \quad \\ \quad & \quad & \quad \end{pmatrix} \begin{pmatrix} \quad \\ \quad \end{pmatrix} + \begin{pmatrix} \quad \\ \quad \end{pmatrix} \\ &= \begin{pmatrix} \quad & \quad & \quad \\ \quad & \quad & \quad \end{pmatrix} + \begin{pmatrix} \quad \\ \quad \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 8 \end{pmatrix} \end{aligned}$$

Alternatively, one can use Python:

```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. (2×2 matrices) Calculate $\boldsymbol{z}$, given:

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

2. (3×3 matrices) Calculate $\boldsymbol{z}$, given:

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

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

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

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

Activation Functions

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

- Sigmoid: $S(z_i) = \frac{1}{1+\mathrm{e}^{-z_i}}$, constrains each value in $\boldsymbol{z}$ to $(0, 1)$
- Tanh: $\tanh(z_i) = \frac{\mathrm{e}^{2z_i}-1}{\mathrm{e}^{2z_i}+1}$, constrains each value in $\boldsymbol{z}$ to $(-1, 1)$.
- ReLU: $\mathrm{ReLU}(z_i) = \max(0, z_i)$, only activates for a value of $\boldsymbol{z}$ if it is positive.
- Softmax: $\sigma(z_i) = \frac{\mathrm{e}^{z_i}}{\Sigma_{j=1}^{K}\mathrm{e}^{z_j}}$. This maps the values in $\boldsymbol{z}$ 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 $\boldsymbol{z} = \begin{pmatrix} 1 \\ 8 \end{pmatrix}$, calculate the resulting vector $\boldsymbol{a} = \text{activation}(\boldsymbol{z})$ using the four activation functions above.

- Sigmoid:

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

- Tanh:

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

- ReLU

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

- Softmax

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

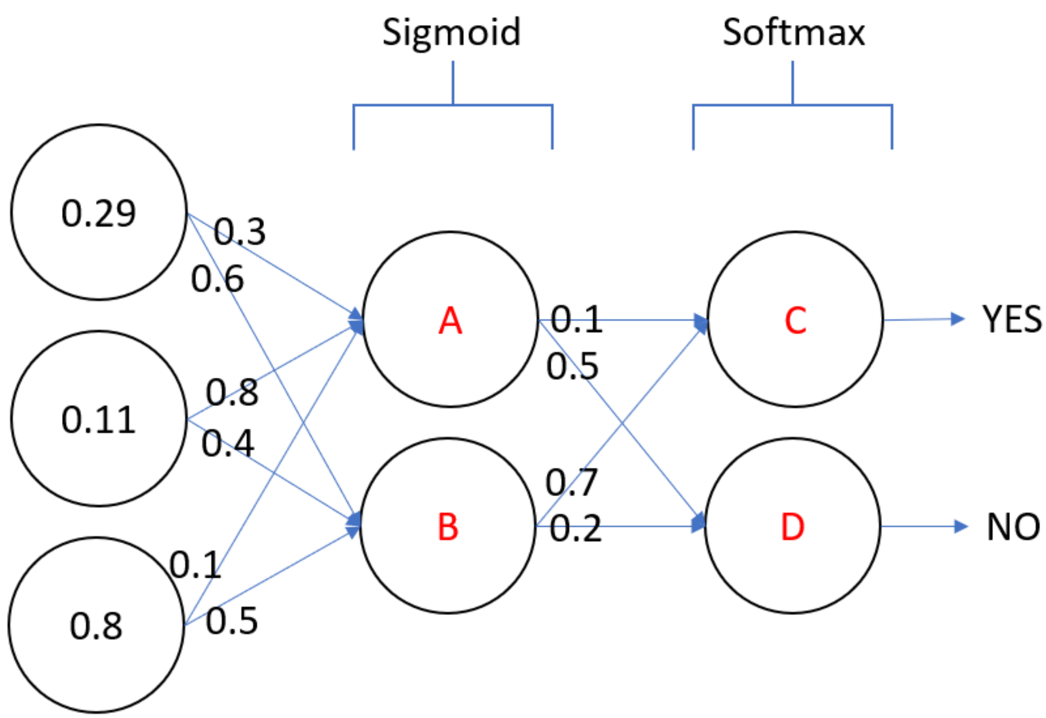
Exercises

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

3. For extra practice, try calculating the vector $\boldsymbol{a}$, 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 $\boldsymbol{X}$ (should be 1×3):

$$\boldsymbol{X} = \begin{pmatrix} \quad\quad\quad\quad\quad \end{pmatrix}.$$

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

$$\boldsymbol{w}_1 = \begin{pmatrix} \\ \\ \\ \end{pmatrix}, \boldsymbol{b}_1 = \begin{pmatrix} \quad\quad\quad\quad \end{pmatrix}.$$

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

$$\boldsymbol{w}_2 = \begin{pmatrix} \\ \\ \end{pmatrix}, \boldsymbol{b}_2 = \begin{pmatrix} \quad\quad\quad\quad \end{pmatrix}.$$

See more details in maths-of-neural-networks.ipynb.