

✔ Congratulations! You passed!

Grade received 100% To pass 80% or higher

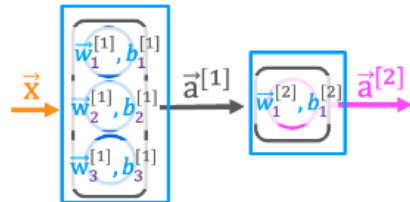
[Go to next item](#)

Neural network implementation in Python

Latest Submission Grade 100%

1. forward prop (coffee roasting model)

1 / 1 point



```


$$a_1^{[2]} = g(\bar{w}_1^{[2]} \cdot \bar{a}^{[1]} + b_1^{[2]})$$

w2_1 = np.array([-7, 8])
b2_1 = np.array([3])
z2_1 = np.dot(w2_1, a1) + b2_1
a2_1 = sigmoid(z2_1)
    
```

```
x = np.array([200, 17])
```

```


$$a_1^{[1]} = g(\bar{w}_1^{[1]} \cdot \bar{x} + b_1^{[1]}) \quad a_2^{[1]} = g(\bar{w}_2^{[1]} \cdot \bar{x} + b_2^{[1]}) \quad a_3^{[1]} = g(\bar{w}_3^{[1]} \cdot \bar{x} + b_3^{[1]})$$

```

```

w1_1 = np.array([1, 2])   w1_2 = np.array([-3, 4])   w1_3 = np.array([5, -6])
b1_1 = np.array([-1])    b1_2 = np.array([1])    b1_3 = np.array([2])
z1_1 = np.dot(w1_1, x) + b1_1   z1_2 = np.dot(w1_2, x) + b1_2   z1_3 = ?
a1_1 = sigmoid(z1_1)   a1_2 = sigmoid(z1_2)   a1_3 = ?
a1 = np.array([a1_1, a1_2, a1_3])
    
```

According to the lecture, how do you calculate the activation of the third neuron in the first layer using NumPy?

☐

```
layer_1 = Dense(units=3, activation='sigmoid')
```

```
a_1 = layer_1(x)
```

☐

```
z1_3 = w1_3 * x + b
```

```
a1_3 = sigmoid(z1_3)
```

☐



```
z1_3 = np.dot(w1_3, x) + b
```

```
a1_3 = sigmoid(z1_3)
```



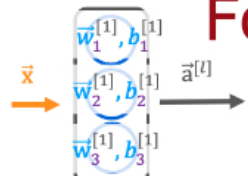
Correct

Correct. Use the numpy.dot function to take the dot product. The sigmoid function shown in lecture can be a function that you write yourself (see course 1, week 3 of this specialization), and that will be provided to you in this course.

2.

1 / 1 point

Forward prop in NumPy



$$\vec{w}_1^{[1]} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \vec{w}_2^{[1]} = \begin{bmatrix} -3 \\ 4 \end{bmatrix} \quad \vec{w}_3^{[1]} = \begin{bmatrix} 5 \\ -6 \end{bmatrix}$$

$$W = \text{np.array}(\begin{bmatrix} 1 & -3 & 5 \\ 2 & 4 & -6 \end{bmatrix}) \quad \text{2 by 3}$$

$$b_1^{[1]} = -1 \quad b_2^{[1]} = 1 \quad b_3^{[1]} = 2$$

$$b = \text{np.array}([-1, 1, 2])$$

$$\vec{a}^{[0]} = \vec{x}$$

$$a_in = \text{np.array}([-2, 4])$$

```
def dense(a_in, W, b, g):
    units = W.shape[1]
    a_out = np.zeros(units)
    for j in range(units):
        w = W[:, j]
        z = np.dot(w, a_in) + b[j]
        a_out[j] = g(z)
    return a_out
```

According to the lecture, when coding up the numpy array W, where would you place the w parameters for each neuron?



In the columns of W.



In the rows of W.



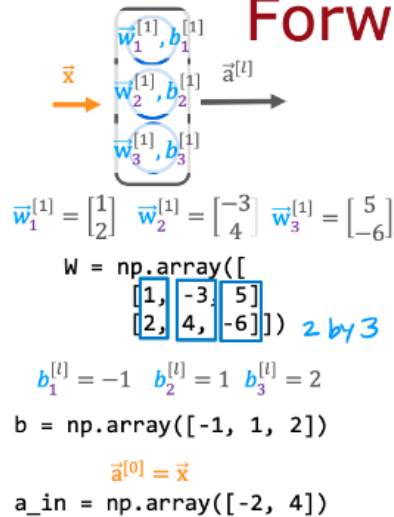
Correct

Correct. The w parameters of neuron 1 are in column 1. The w parameters of neuron 2 are in column 2, and so on.

3.

1 / 1 point

Forward prop in NumPy



```
def dense(a_in, W, b, g):
    units = W.shape[1]
    a_out = np.zeros(units)
    for j in range(units):
        w = W[:, j]
        z = np.dot(w, a_in) + b[j]
        a_out[j] = g(z)
    return a_out
```

For the code above in the "dense" function that defines a single layer of neurons, how many times does the code go through the "for loop"? Note that W has 2 rows and 3 columns.

☐ 6 times

For each neuron in the layer, there is one column in the numpy array W . Each row of W represents how many input features are fed into that layer. The for loop calculates the activation value for each neuron.

☐ 2 times

☒ 3 times

☐ 5 times

For each neuron in the layer, there is one column in the numpy array W . Each row of W represents how many input features are fed into that layer. The for loop calculates the activation value for each neuron.

☒ **Correct**

Yes! For each neuron in the layer, there is one column in the numpy array W . The for loop calculates the activation value for each neuron. So if there are 5 columns in W , there are 5 neurons in the dense layer, and therefore the for loop goes through 5 iterations (one for each neuron).