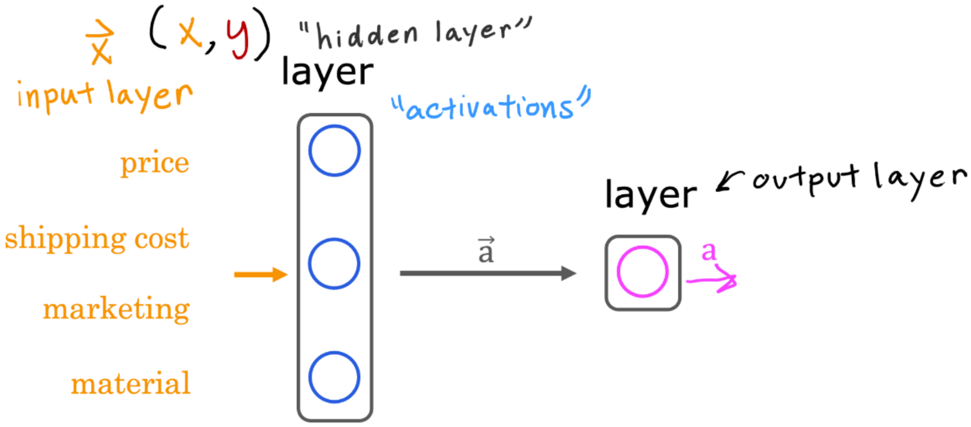
1. 1 point



Which of these are terms used to refer to components of an artificial neural network? (hint: three of these are correct)

☑ neurons

☑ layers

□ axon

☑ activation function

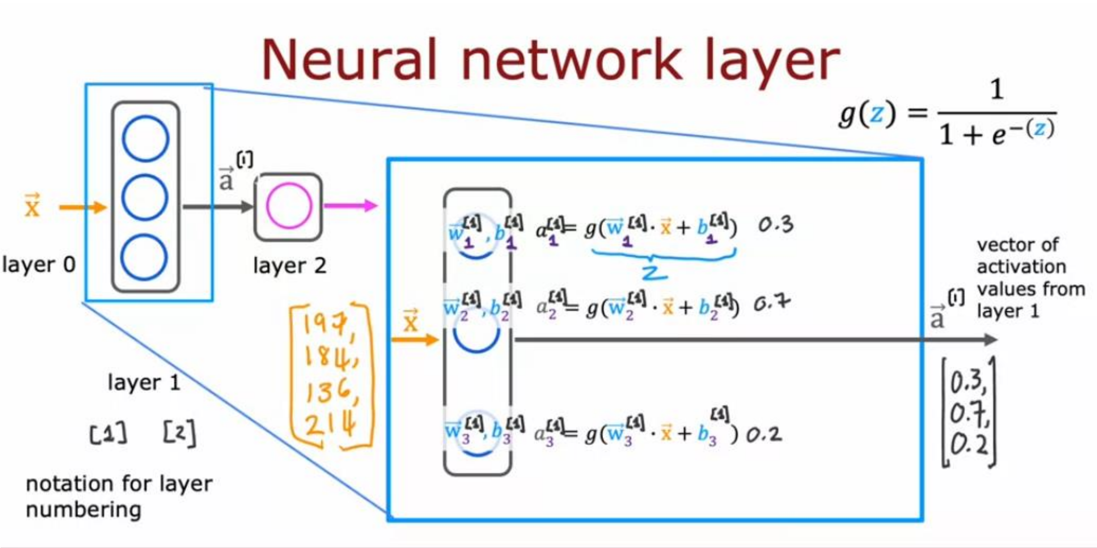
2. True/False? Neural networks take inspiration from, but do not very accurately mimic, how neurons in a biological brain learn. 1 point

● True

○ False

Coursera Honor Code Learn more

☑ I, **Şaban Kara**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.



Neural network layer

$$g(z) = \frac{1}{1+e^{-(z)}}$$

$$a_1^{[1]} = g(\underbrace{\vec{w}_1^{[1]} \cdot \vec{x} + b_1^{[1]}}_{z}) \quad 0.3$$

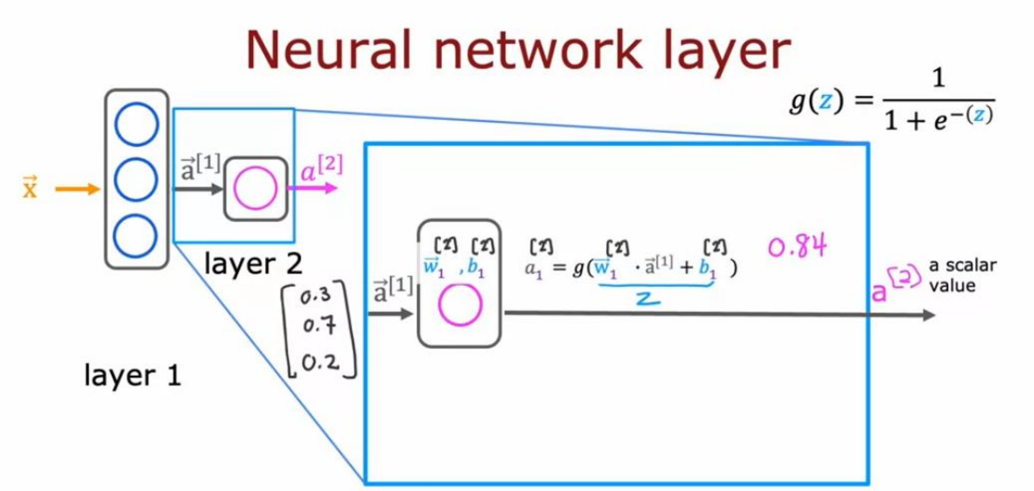
$$a_2^{[1]} = g(\vec{w}_2^{[1]} \cdot \vec{x} + b_2^{[1]}) \quad 0.7$$

$$a_3^{[1]} = g(\vec{w}_3^{[1]} \cdot \vec{x} + b_3^{[1]}) \quad 0.2$$

$\vec{a}^{[1]}$ vector of activation values from layer 1

$\vec{x}$
layer 0
$\vec{a}^{[1]}$
layer 2
layer 1
[1] [2]
notation for layer numbering
$\vec{w}_1^{[1]}, b_1^{[1]}$
$\vec{w}_2^{[1]}, b_2^{[1]}$
$\vec{w}_3^{[1]}, b_3^{[1]}$
[197, 184, 136, 214]
[0.3, 0.7, 0.2]

A üstü parantez sayı hangi katmana ait olduğunu gösterir.

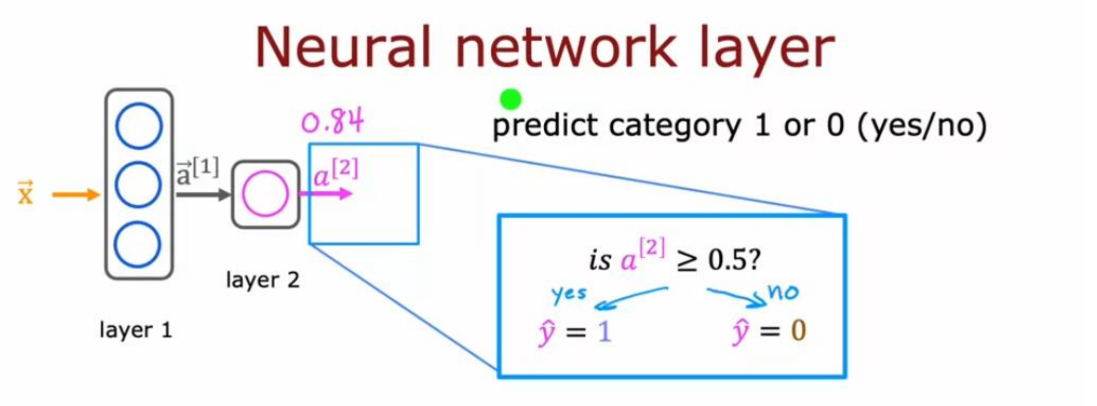


Neural network layer

$$g(z) = \frac{1}{1+e^{-(z)}}$$

$$a_1^{[2]} = g(\underbrace{\vec{w}_1^{[2]} \cdot \vec{a}^{[1]} + b_1^{[2]}}_{z}) \quad 0.84$$

$a^{[2]}$ a scalar value



Neural network layer

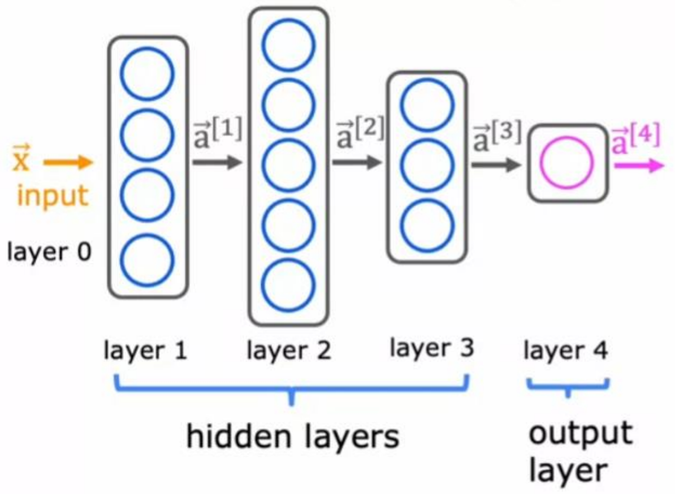
predict category 1 or 0 (yes/no)

is $a^{[2]} \geq 0.5$?

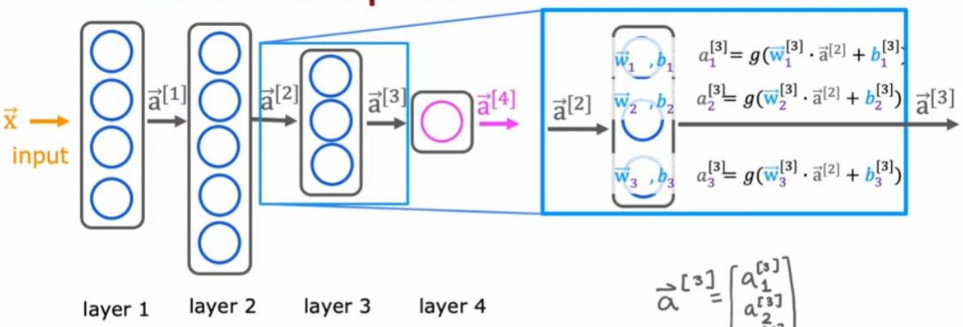
yes $\hat{y} = 1$

no $\hat{y} = 0$

More complex neural network



More complex neural network



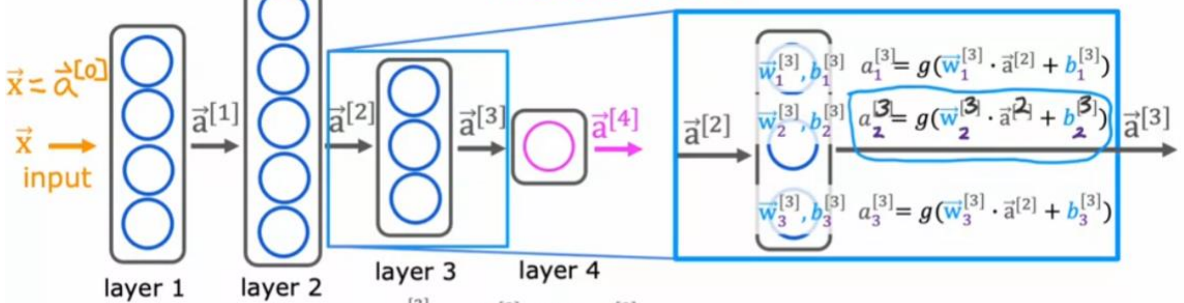
$$a_1^{[3]} = g(\vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]})$$

$$a_2^{[3]} = g(\vec{w}_2^{[3]} \cdot \vec{a}^{[2]} + b_2^{[3]})$$

$$a_3^{[3]} = g(\vec{w}_3^{[3]} \cdot \vec{a}^{[2]} + b_3^{[3]})$$

$$\vec{a}^{[3]} = \begin{bmatrix} a_1^{[3]} \\ a_2^{[3]} \\ a_3^{[3]} \end{bmatrix}$$

Notation



$$a_1^{[3]} = g(\vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]})$$

$$a_2^{[3]} = g(\vec{w}_2^{[3]} \cdot \vec{a}^{[2]} + b_2^{[3]})$$

$$a_3^{[3]} = g(\vec{w}_3^{[3]} \cdot \vec{a}^{[2]} + b_3^{[3]})$$

$$a_2^{[3]} = g(\vec{w}_2^{[3]} \cdot \vec{a}^{[2]} + b_2^{[3]})$$

Activation value of layer l, unit(neuron) j

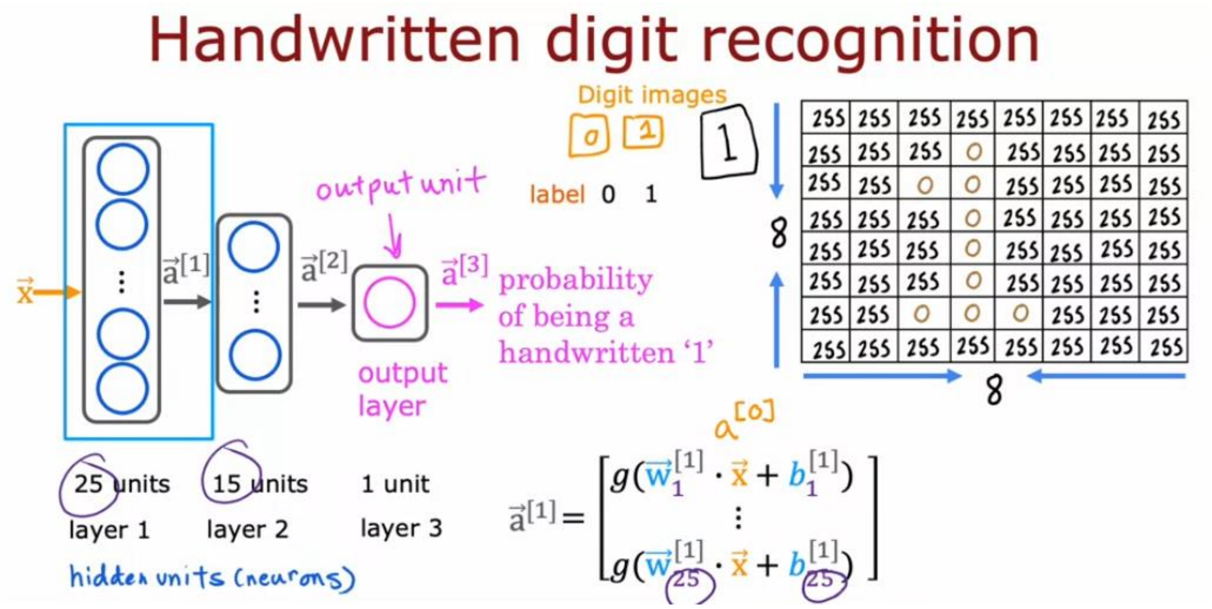
output of layer $l-1$ (previous layer)

$$\boxed{a_j^{[l]} = g(\vec{w}_j^{[l]} \cdot \vec{a}^{[l-1]} + b_j^{[l]})}$$

sigmoid

"activation function"

Parameters w & b of layer l, unit j



Handwritten digit recognition

Digit images

0 1

label 0 1

output unit

$\vec{x}$ $\vec{a}^{[1]}$ $\vec{a}^{[2]}$ $\vec{a}^{[3]}$

probability of being a handwritten '1'

output layer

255	255	255	255	255	255	255	255
255	255	255	0	255	255	255	255
255	255	0	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	0	0	0	255	255	255
255	255	255	255	255	255	255	255

8

8

25 units layer 1

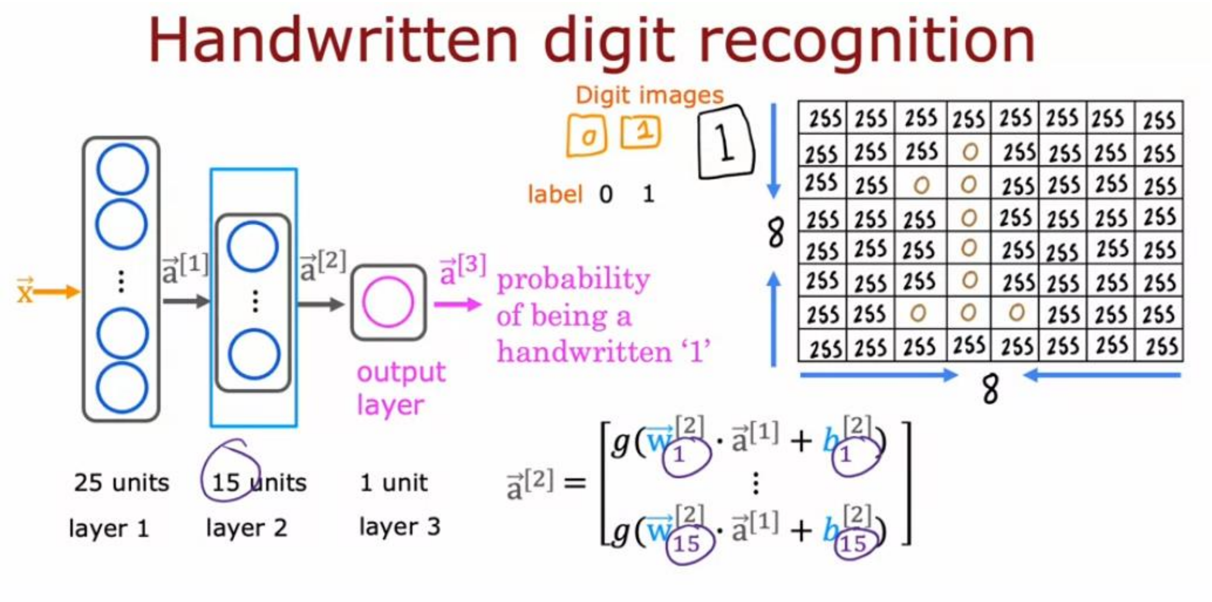
15 units layer 2

1 unit layer 3

hidden units (neurons)

$a^{[0]}$

$$\vec{a}^{[1]} = \begin{bmatrix} g(\vec{w}_1^{[1]} \cdot \vec{x} + b_1^{[1]}) \\ \vdots \\ g(\vec{w}_{25}^{[1]} \cdot \vec{x} + b_{25}^{[1]}) \end{bmatrix}$$



Handwritten digit recognition

Digit images

0 1

label 0 1

$\vec{x}$ $\vec{a}^{[1]}$ $\vec{a}^{[2]}$ $\vec{a}^{[3]}$

probability of being a handwritten '1'

output layer

255	255	255	255	255	255	255	255
255	255	255	0	255	255	255	255
255	255	0	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	0	0	0	255	255	255
255	255	255	255	255	255	255	255

8

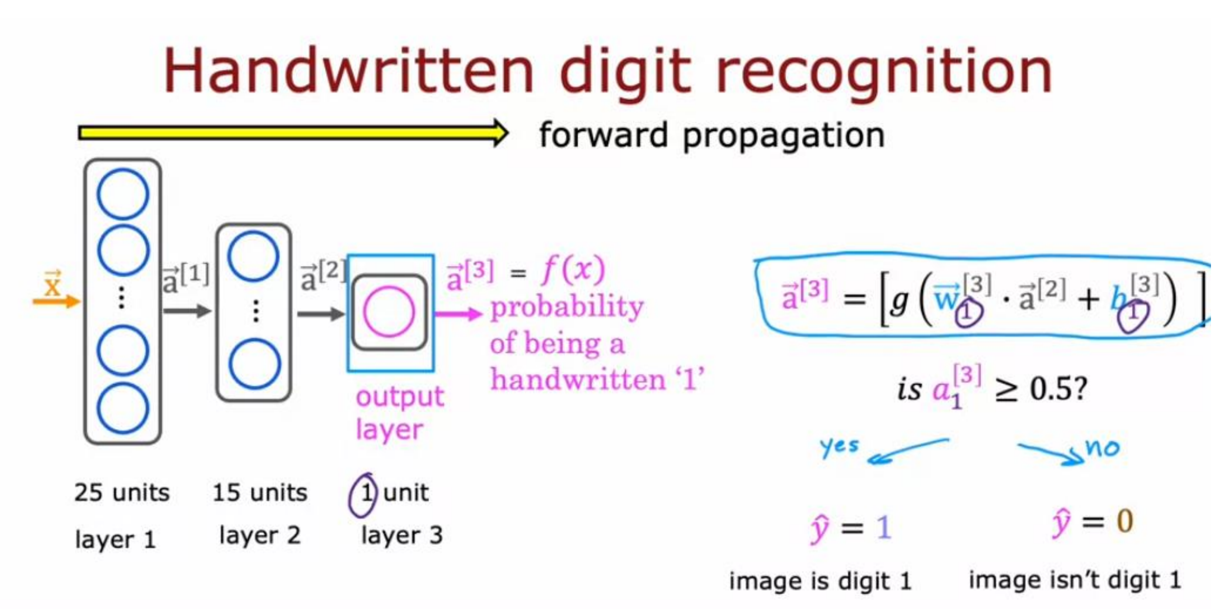
8

25 units layer 1

15 units layer 2

1 unit layer 3

$$\vec{a}^{[2]} = \begin{bmatrix} g(\vec{w}_1^{[2]} \cdot \vec{a}^{[1]} + b_1^{[2]}) \\ \vdots \\ g(\vec{w}_{15}^{[2]} \cdot \vec{a}^{[1]} + b_{15}^{[2]}) \end{bmatrix}$$



Handwritten digit recognition

forward propagation

$\vec{x}$ $\vec{a}^{[1]}$ $\vec{a}^{[2]}$

$\vec{a}^{[3]} = f(x)$

probability of being a handwritten '1'

output layer

25 units layer 1

15 units layer 2

1 unit layer 3

$$\vec{a}^{[3]} = \left[g\left(\vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]}\right) \right]$$

is $a_1^{[3]} \geq 0.5$?

yes no

$\hat{y} = 1$

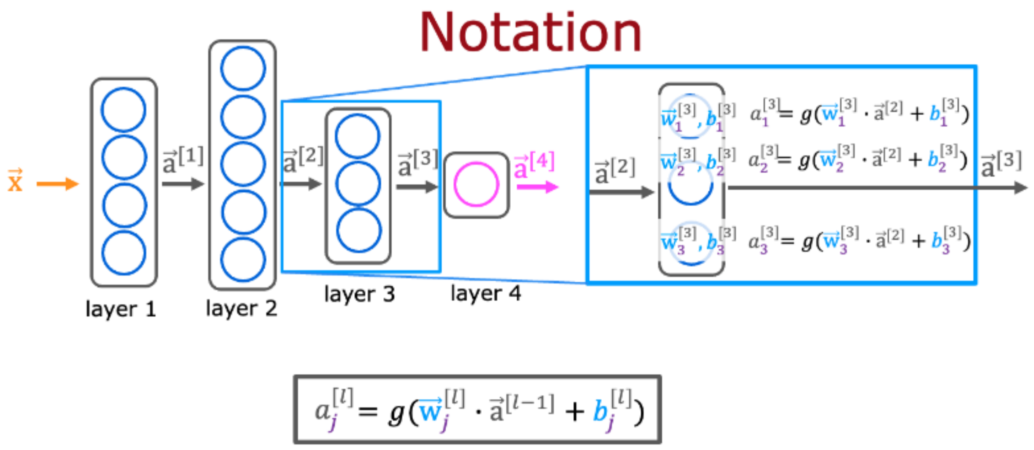
image is digit 1

$\hat{y} = 0$

image isn't digit 1

1.

1 point


Notation
$\vec{x}$
layer 1
$\vec{a}^{[1]}$
layer 2
$\vec{a}^{[2]}$
layer 3
$\vec{a}^{[3]}$
layer 4
$\vec{a}^{[4]}$
$\vec{w}_1^{[3]}, b_1^{[3]}$
$\vec{w}_2^{[3]}, b_2^{[3]}$
$\vec{w}_3^{[3]}, b_3^{[3]}$
$a_1^{[3]} = g(\vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]})$
$a_2^{[3]} = g(\vec{w}_2^{[3]} \cdot \vec{a}^{[2]} + b_2^{[3]})$
$a_3^{[3]} = g(\vec{w}_3^{[3]} \cdot \vec{a}^{[2]} + b_3^{[3]})$

$$\boxed{a_j^{[l]} = g(\vec{w}_j^{[l]} \cdot \vec{a}^{[l-1]} + b_j^{[l]})}$$

For a neural network, what is the expression for calculating the activation of the third neuron in layer 2? Note, this is different from the question that you saw in the lecture video.

○ $a_3^{[2]} = g(\vec{w}_3^{[2]} \cdot \vec{a}^{[2]} + b_3^{[2]})$

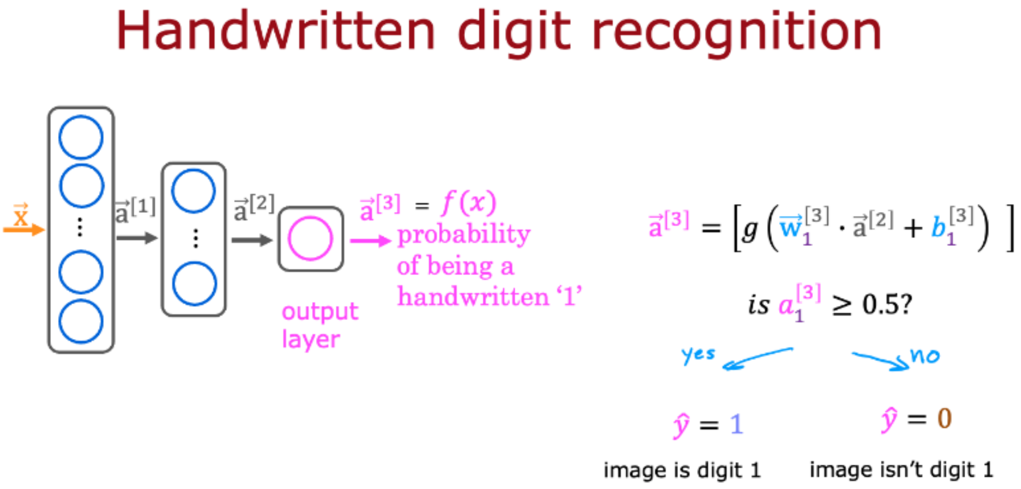
● $a_3^{[2]} = g(\vec{w}_3^{[2]} \cdot \vec{a}^{[1]} + b_3^{[2]})$

○ $a_3^{[2]} = g(\vec{w}_2^{[3]} \cdot \vec{a}^{[2]} + b_2^{[3]})$

○ $a_3^{[2]} = g(\vec{w}_2^{[3]} \cdot \vec{a}^{[1]} + b_2^{[3]})$

2.

1 point



For the handwriting recognition task discussed in lecture, what is the output $a_1^{[3]}$?

○ A vector of several numbers, each of which is either exactly 0 or 1

○ A vector of several numbers that take values between 0 and 1

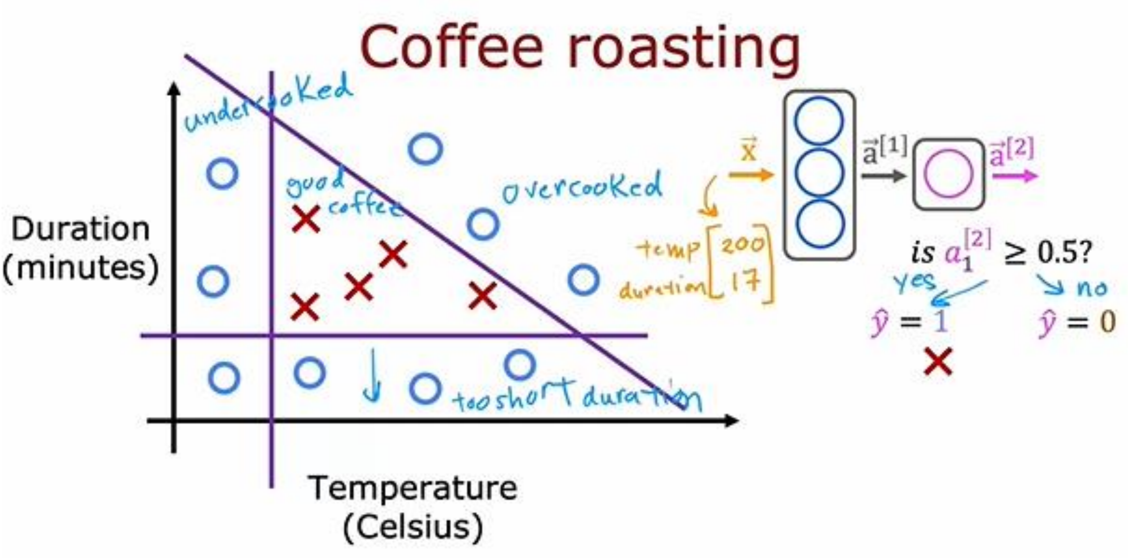
○ A number that is either exactly 0 or 1, comprising the network's prediction

● The estimated probability that the input image is of a number 1, a number that ranges from 0 to 1.

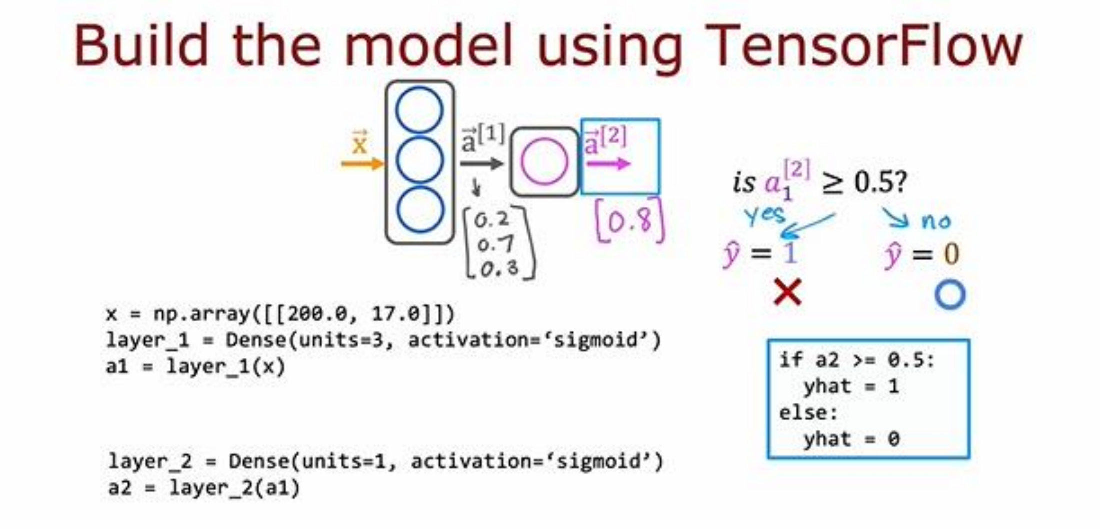
Coursera Honor Code Learn more

□ I, **Şaban Kara**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Inference in Code



Coffee roasting

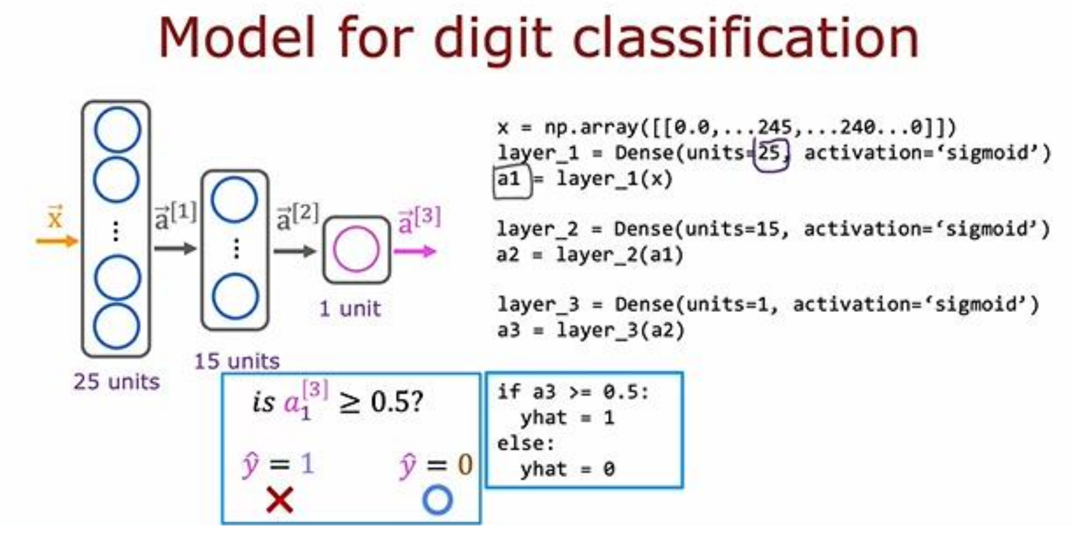


Build the model using TensorFlow

```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation='sigmoid')
a1 = layer_1(x)


layer_2 = Dense(units=1, activation='sigmoid')
a2 = layer_2(a1)
```

```
if a2 >= 0.5:
  yhat = 1
else:
  yhat = 0
```



Model for digit classification

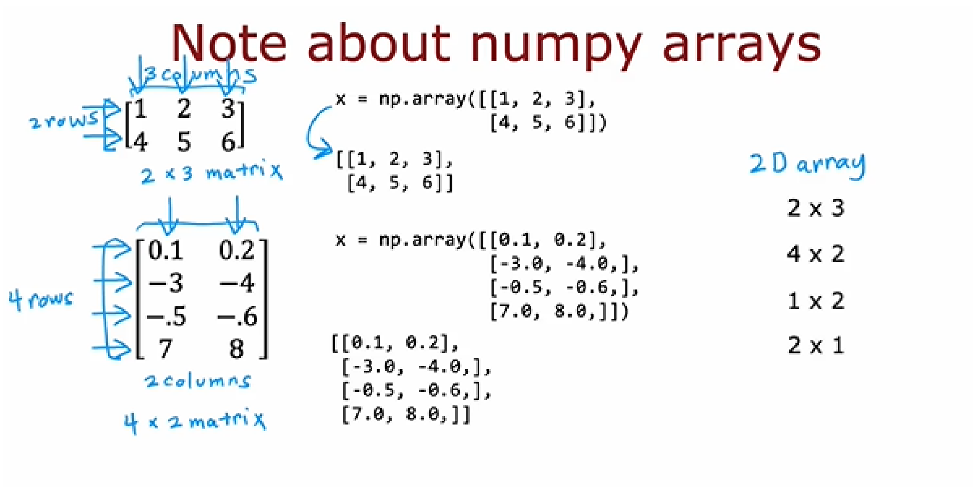
```
x = np.array([[0.0,...245,...240...0]])
layer_1 = Dense(units=25, activation='sigmoid')
a1 = layer_1(x)

layer_2 = Dense(units=15, activation='sigmoid')
a2 = layer_2(a1)

layer_3 = Dense(units=1, activation='sigmoid')
a3 = layer_3(a2)
```

```
if a3 >= 0.5:
  yhat = 1
else:
  yhat = 0
```

Data in Tensorflow



Note about numpy arrays

3 columns

2 rows $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

2 x 3 matrix

```
x = np.array([[1, 2, 3],
              [4, 5, 6]])
[[1, 2, 3],
 [4, 5, 6]]
```

2D array

2 x 3

4 x 2

1 x 2

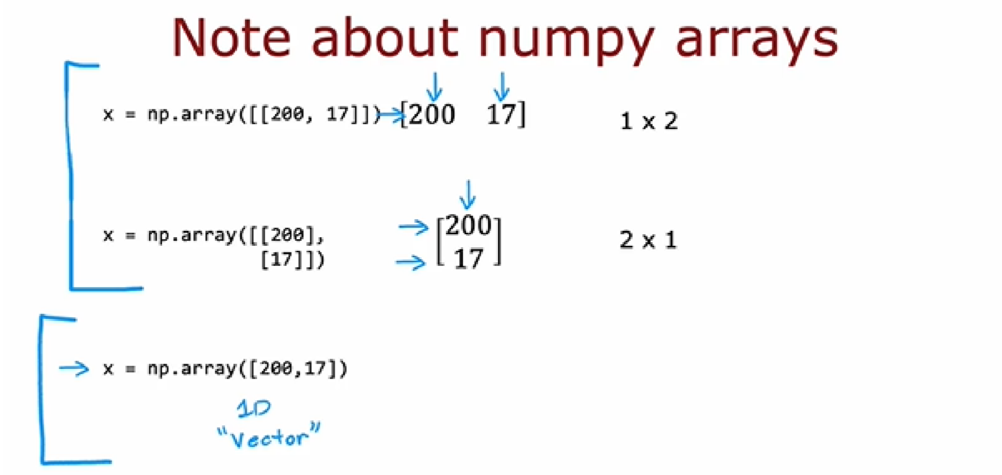
2 x 1

4 rows $\begin{bmatrix} 0.1 & 0.2 \\ -3 & -4 \\ -.5 & -.6 \\ 7 & 8 \end{bmatrix}$

2 columns

4 x 2 matrix

```
x = np.array([[0.1, 0.2],
              [-3.0, -4.0,],
              [-0.5, -0.6,],
              [7.0, 8.0,]])
[[0.1, 0.2],
 [-3.0, -4.0,],
 [-0.5, -0.6,],
 [7.0, 8.0,]]
```



Note about numpy arrays

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

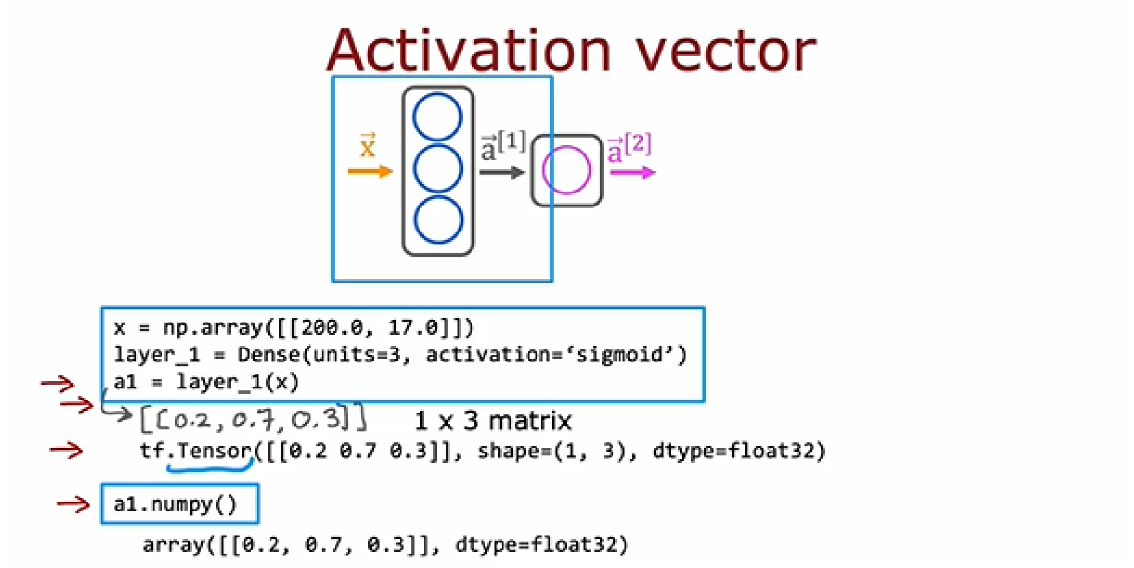
$\rightarrow [200 \quad 17]$ 1 x 2

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

$\begin{bmatrix} 200 \\ 17 \end{bmatrix}$ 2 x 1

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

1D "Vector"



Activation vector

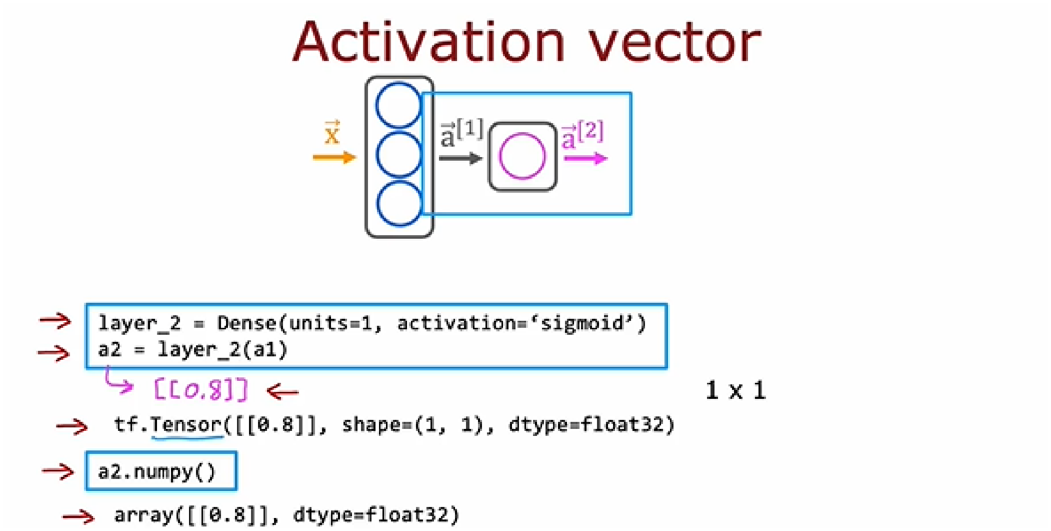
```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation='sigmoid')
a1 = layer_1(x)
```

[[0.2, 0.7, 0.3]]   1 x 3 matrix

```
tf.Tensor([[0.2 0.7 0.3]], shape=(1, 3), dtype=float32)
```

```
a1.numpy()
```

```
array([[0.2, 0.7, 0.3]], dtype=float32)
```



Activation vector

```
layer_2 = Dense(units=1, activation='sigmoid')
a2 = layer_2(a1)
```

[[0.8]]

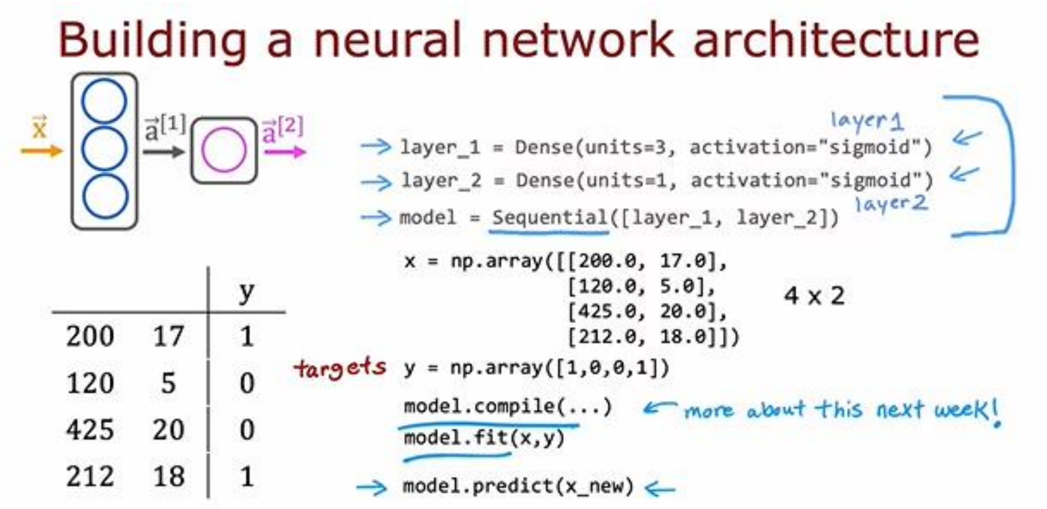
1 x 1

```
tf.Tensor([[0.8]], shape=(1, 1), dtype=float32)
```

```
a2.numpy()
```

```
array([[0.8]], dtype=float32)
```

Building a Neural Network



Building a neural network architecture

```
layer_1 = Dense(units=3, activation="sigmoid")
layer_2 = Dense(units=1, activation="sigmoid")
model = Sequential([layer_1, layer_2])
```

layer1

layer2

		y
200	17	1
120	5	0
425	20	0
212	18	1

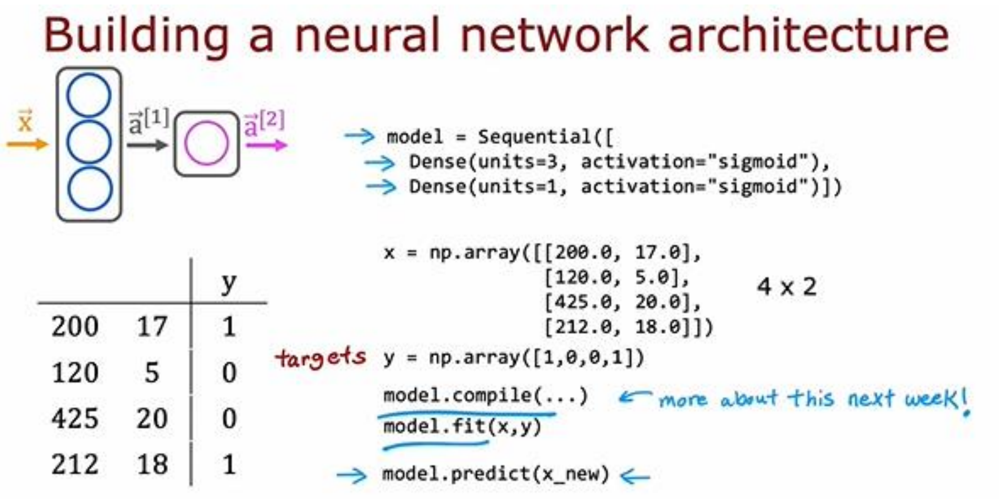
```
x = np.array([[200.0, 17.0],
              [120.0, 5.0],
              [425.0, 20.0],
              [212.0, 18.0]])
```

4 x 2

targets

```
y = np.array([1,0,0,1])
model.compile(...)
model.fit(x,y)
model.predict(x_new)
```

more about this next week!



Building a neural network architecture

```
model = Sequential([
  Dense(units=3, activation="sigmoid"),
  Dense(units=1, activation="sigmoid")])
```

		y
200	17	1
120	5	0
425	20	0
212	18	1

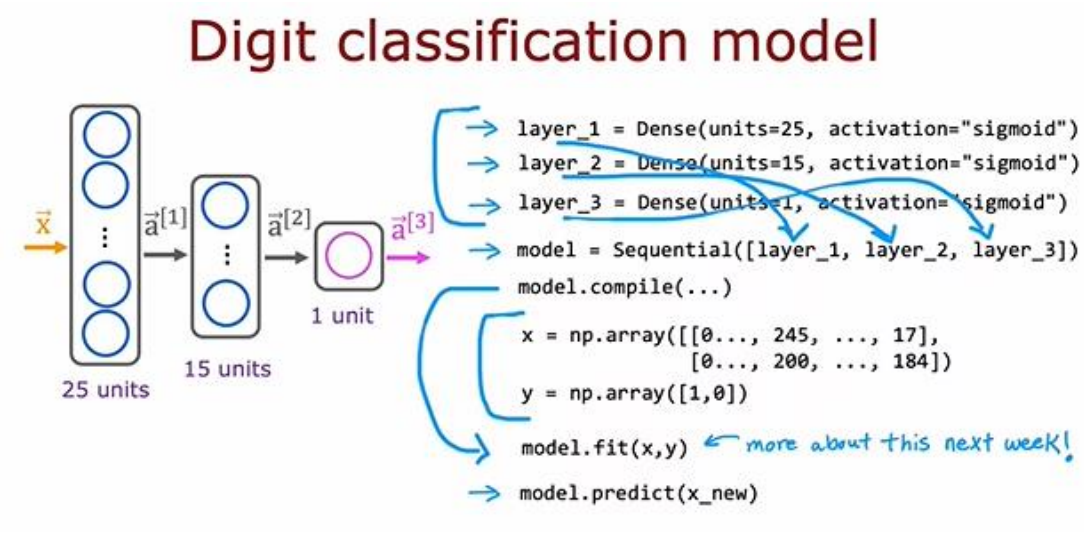
```
x = np.array([[200.0, 17.0],
              [120.0, 5.0],
              [425.0, 20.0],
              [212.0, 18.0]])
```

4 x 2

targets

```
y = np.array([1,0,0,1])
model.compile(...)
model.fit(x,y)
model.predict(x_new)
```

more about this next week!



Digit classification model

```
layer_1 = Dense(units=25, activation="sigmoid")
layer_2 = Dense(units=15, activation="sigmoid")
layer_3 = Dense(units=1, activation="sigmoid")
model = Sequential([layer_1, layer_2, layer_3])
model.compile(...)
 x = np.array([[0..., 245, ..., 17],
               [0..., 200, ..., 184])
 y = np.array([1,0])
model.fit(x,y)
model.predict(x_new)
```

more about this next week!

1. For the the following code:

   1 point

```
model = Sequential([
 Dense(units=25, activation="sigmoid"),
 Dense(units=15, activation="sigmoid"),
 Dense(units=10, activation="sigmoid"),
 Dense(units=1, activation="sigmoid")])
```

This code will define a neural network with how many layers?

○ 3

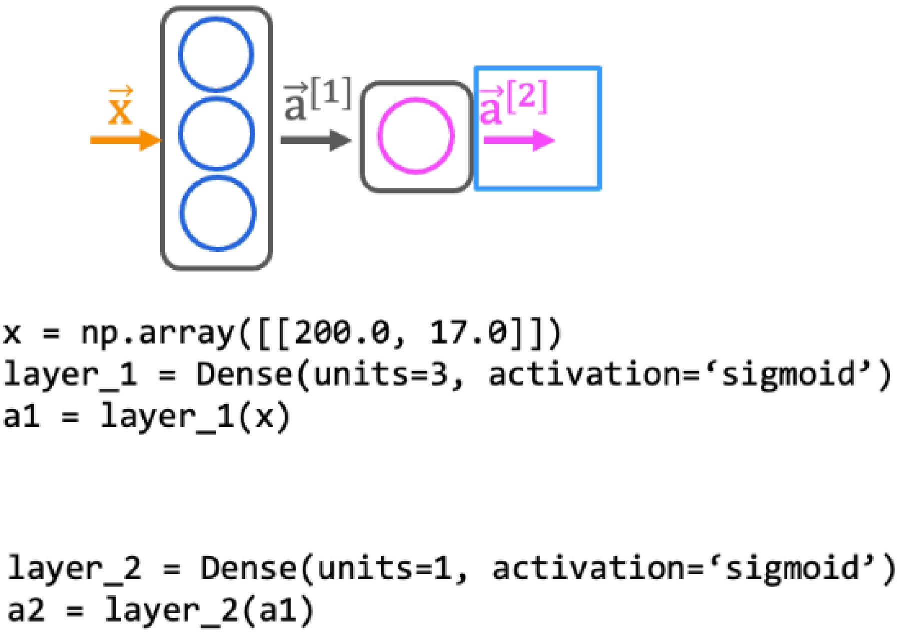
● 4

○ 25

○ 5

2.

   1 point



```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation='sigmoid')
a1 = layer_1(x)


layer_2 = Dense(units=1, activation='sigmoid')
a2 = layer_2(a1)
```

How do you define the second layer of a neural network that has 4 neurons and a sigmoid activation?

○ Dense(units=[4], activation=['sigmoid'])

○ Dense(units=4)

● Dense(units=4, activation='sigmoid')

○ Dense(layer=2, units=4, activation = 'sigmoid')