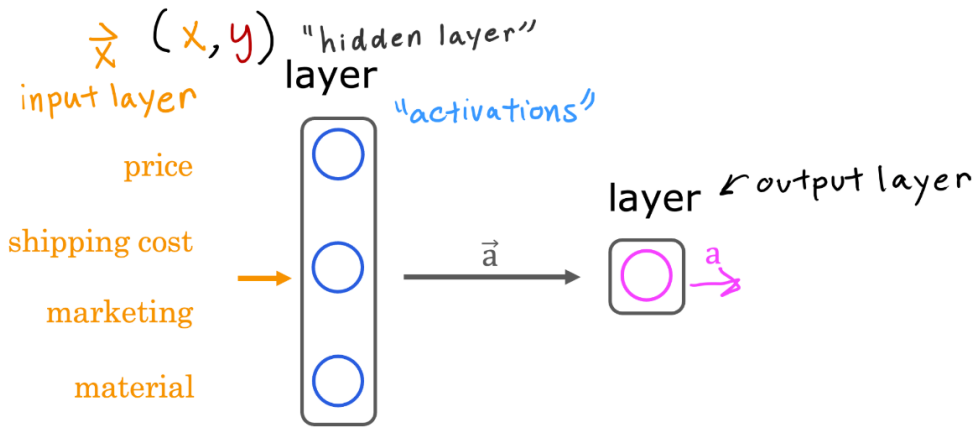
1. 1 point



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

- [x] neurons
- [x] layers
- [ ] axon
- [x] activation function

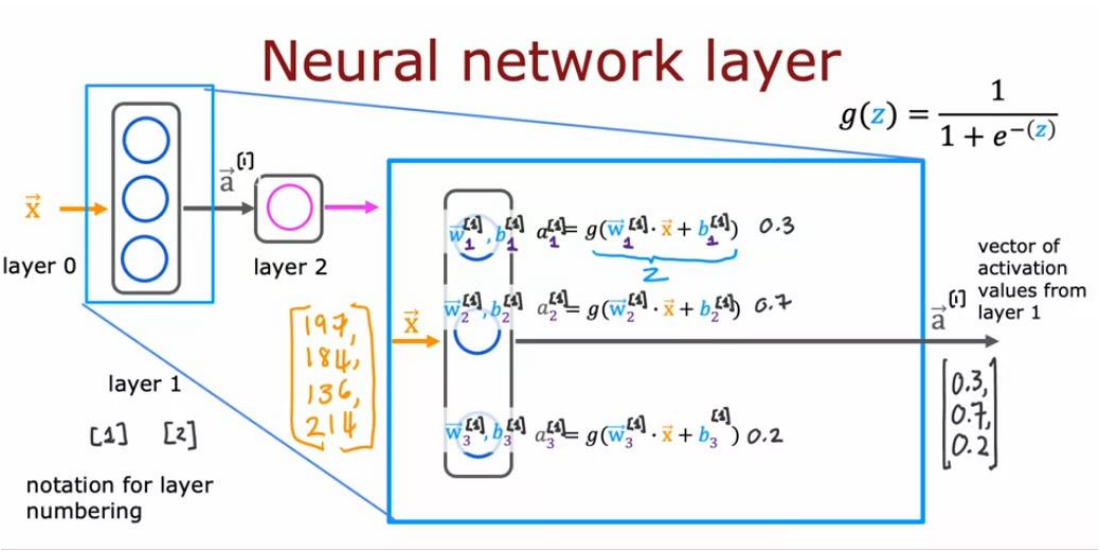
2. True/False? Neural networks take inspiration from, but do not very accurately mimic, how neurons in a biological brain learn. 1 point

- (•) True
- ( ) False

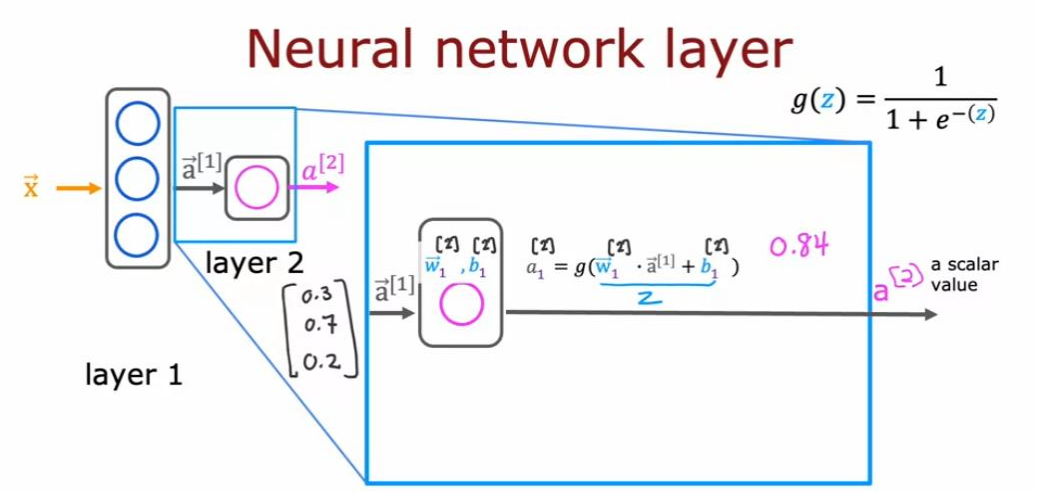
Coursera Honor Code Learn more

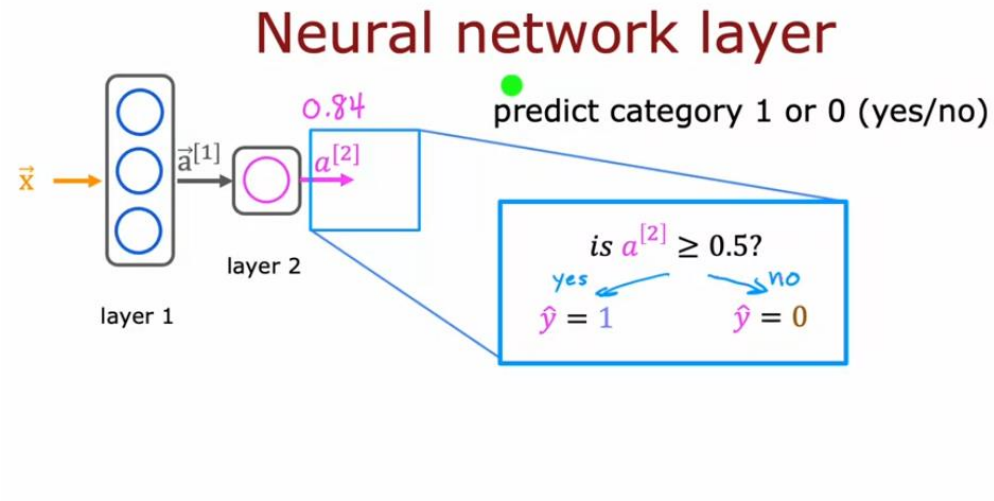
I, **Şaban Kara**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Neural Network Layer

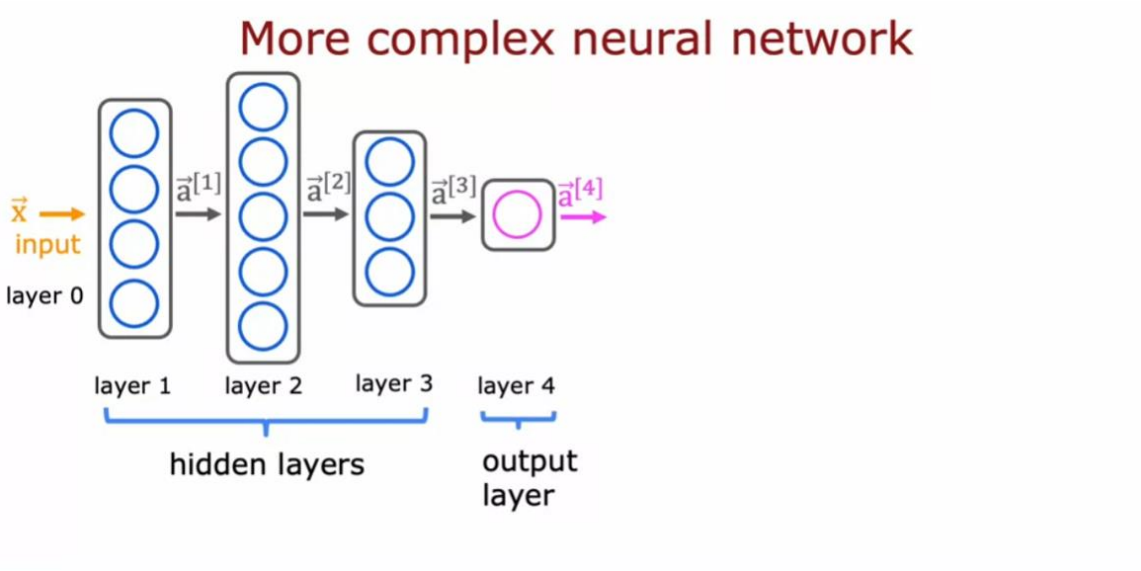


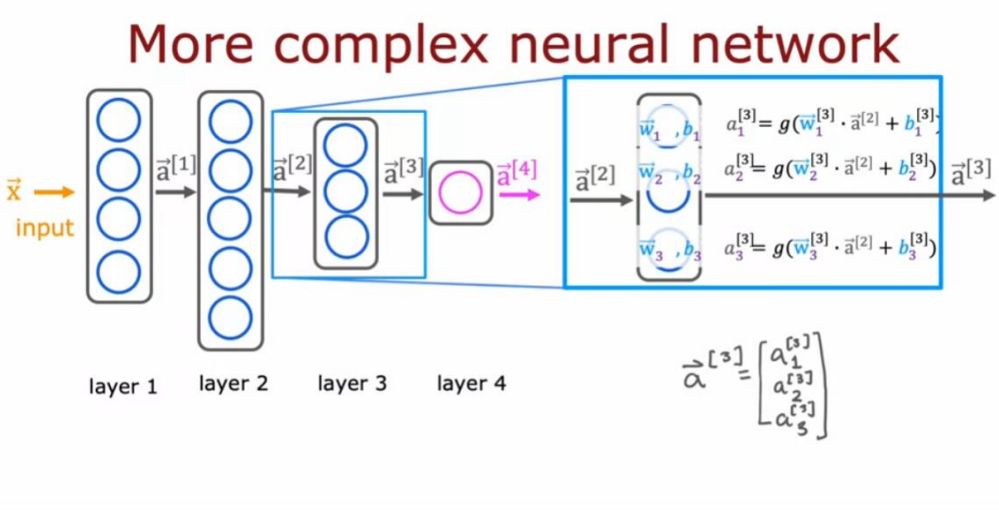
A üstü parantez sayı hangi katmana ait olduğunu gösterir.

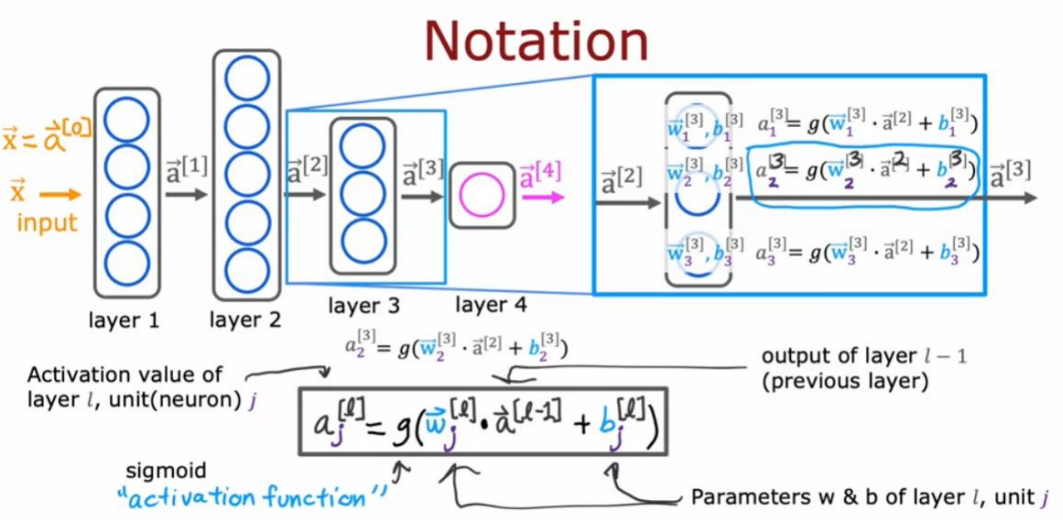




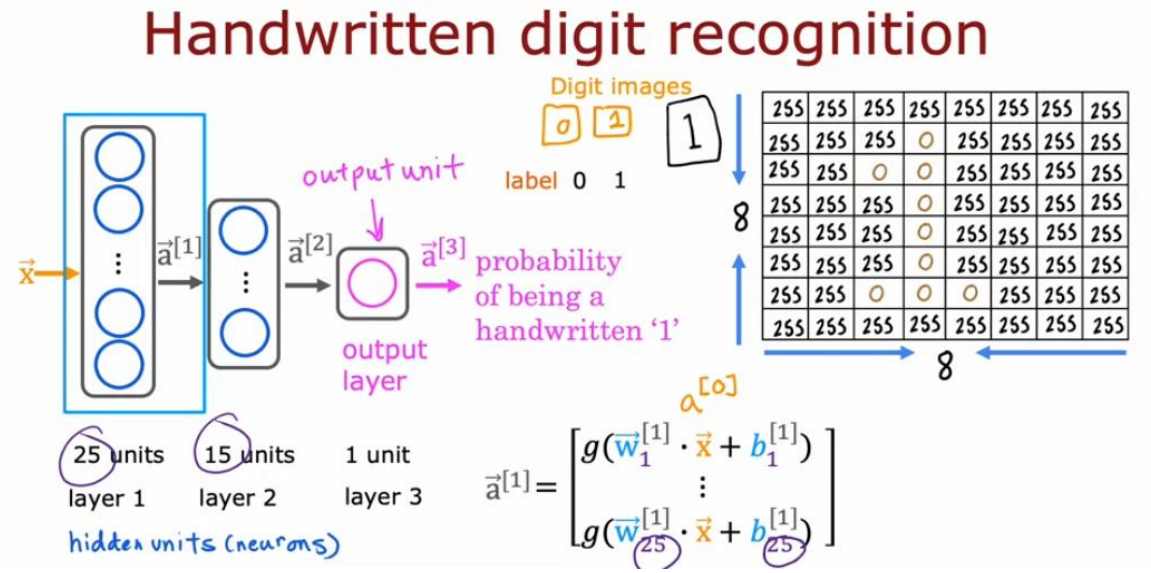
More Complex Neural Networks

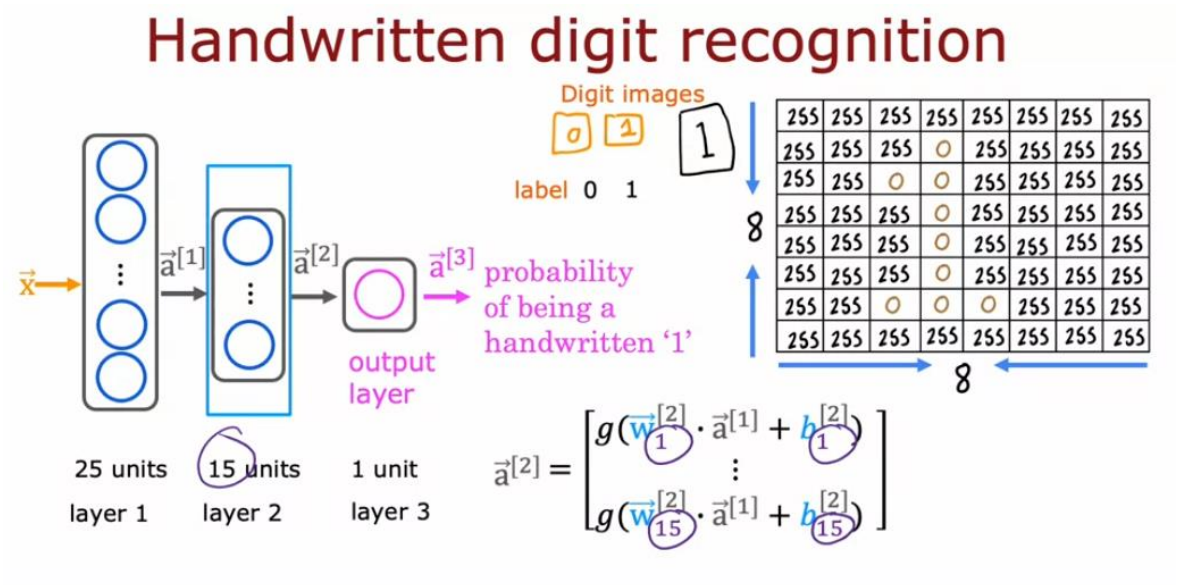


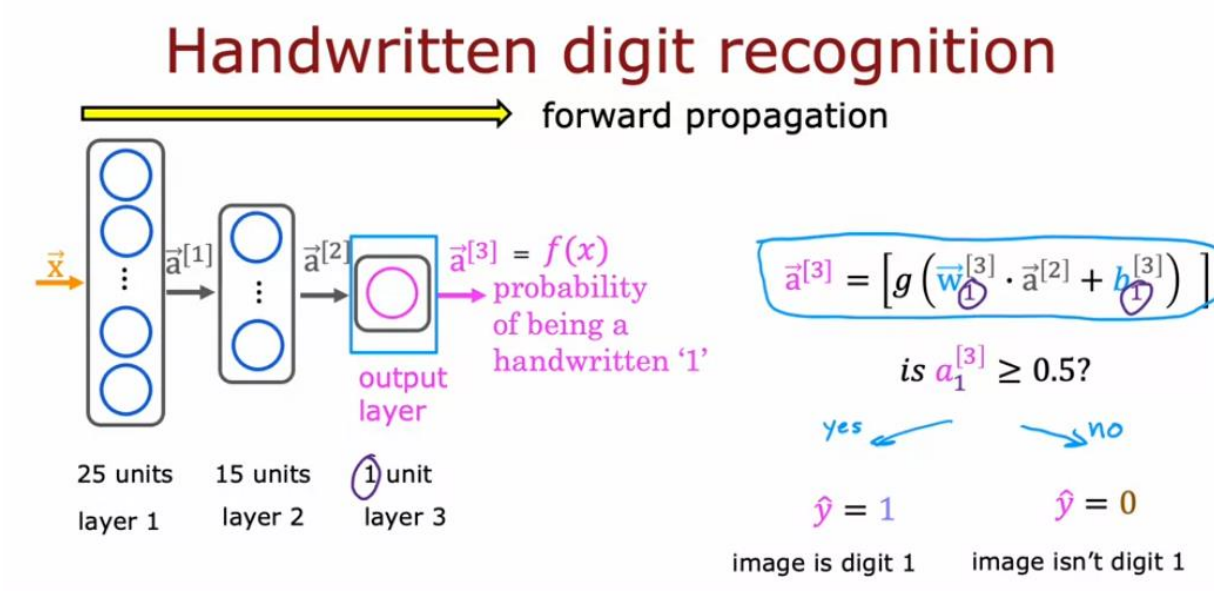




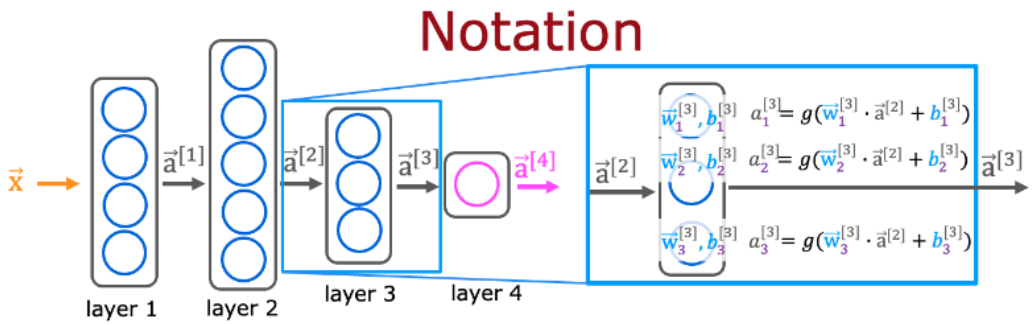
Inference: making predictions (forward propagation)







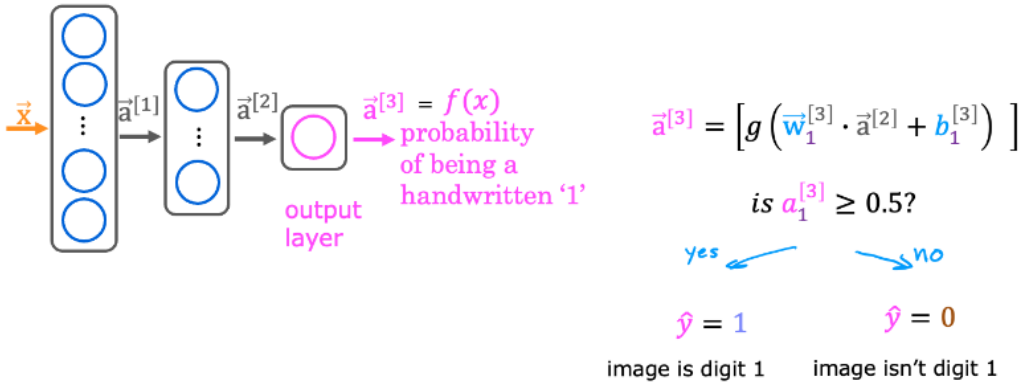
1. 1 point



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.

- $a_3^{[2]} = g(\vec{w}_3^{[2]} \cdot \vec{a}^{[2]} + b_3^{[2]})$
- (selected) $a_3^{[2]} = g(\vec{w}_3^{[2]} \cdot \vec{a}^{[1]} + b_3^{[2]})$
- $a_3^{[2]} = g(\vec{w}_2^{[3]} \cdot \vec{a}^{[2]} + b_2^{[3]})$
- $a_3^{[2]} = g(\vec{w}_2^{[3]} \cdot \vec{a}^{[1]} + b_2^{[3]})$

2. 1 point



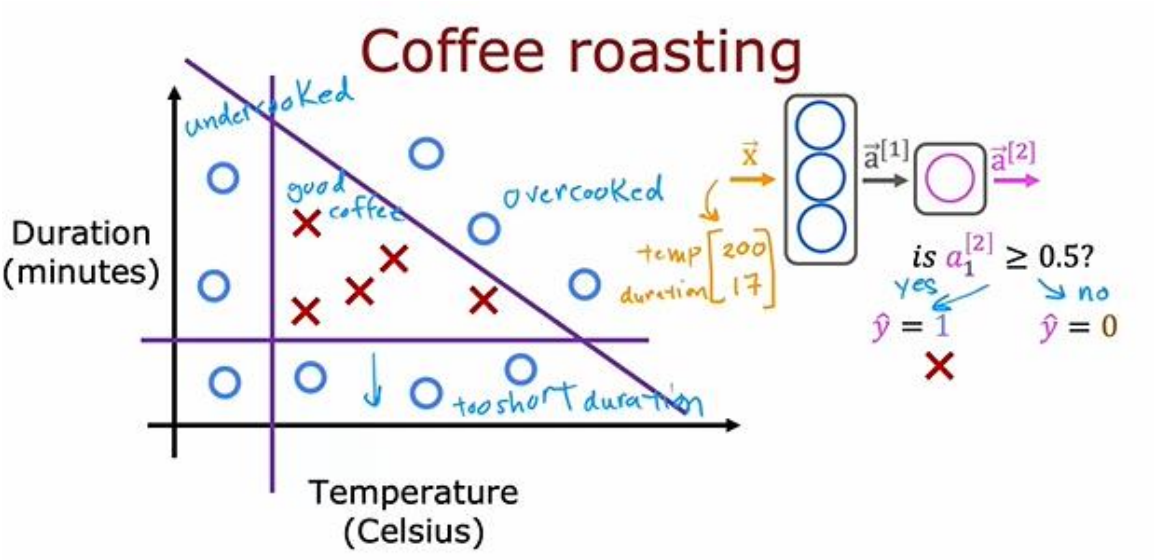
For the handwriting recognition task discussed in lecture, what is the output $a_1^{[3]}$?

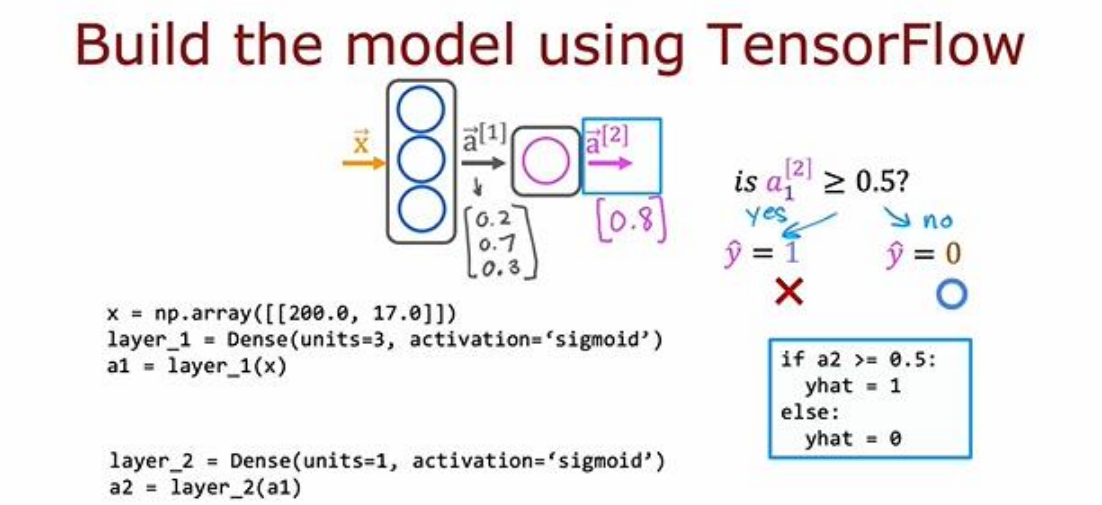
- A vector of several numbers, each of which is either exactly 0 or 1
- A vector of several numbers that take values between 0 and 1
- A number that is either exactly 0 or 1, comprising the network's prediction
- (selected) The estimated probability that the input image is of a number 1, a number that ranges from 0 to 1.

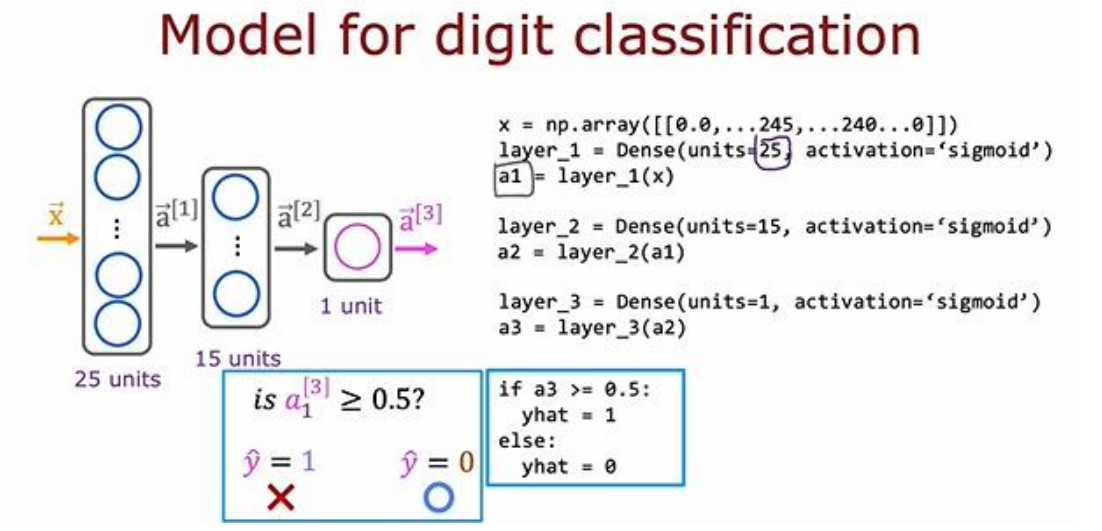
Coursera Honor Code Learn more

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.

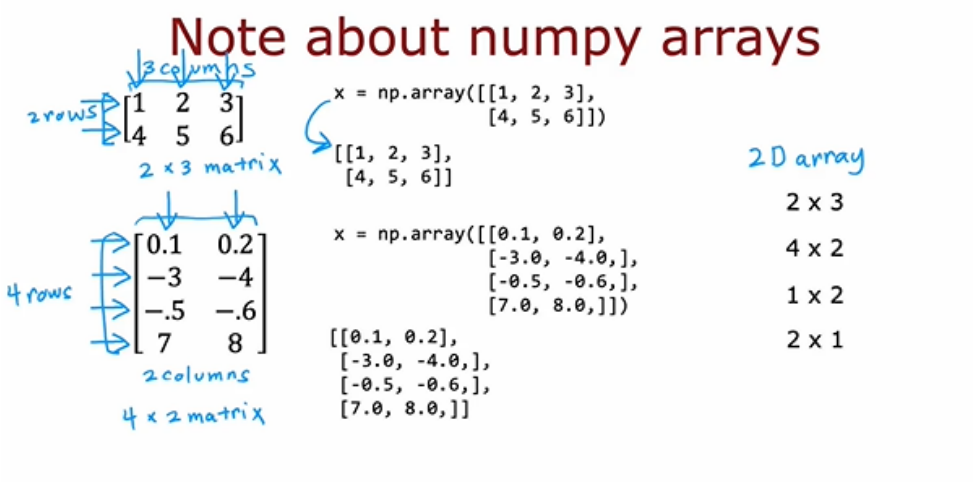
Inference in Code

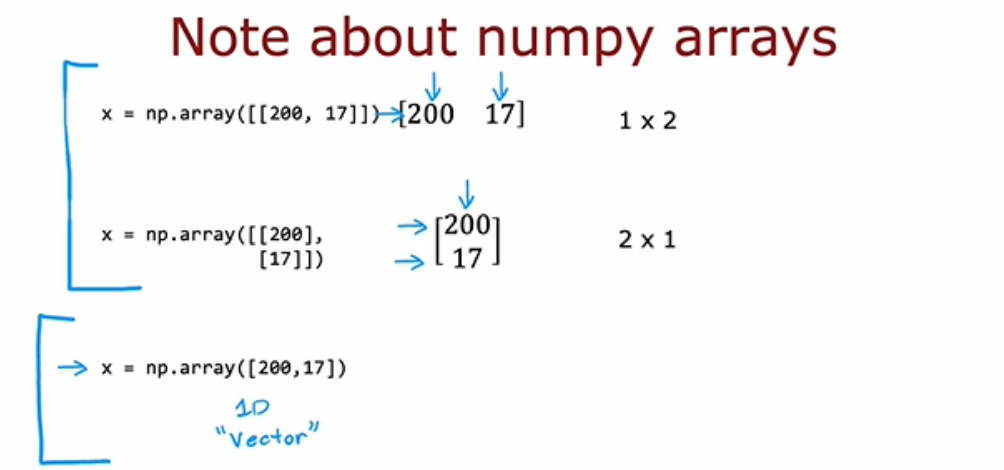


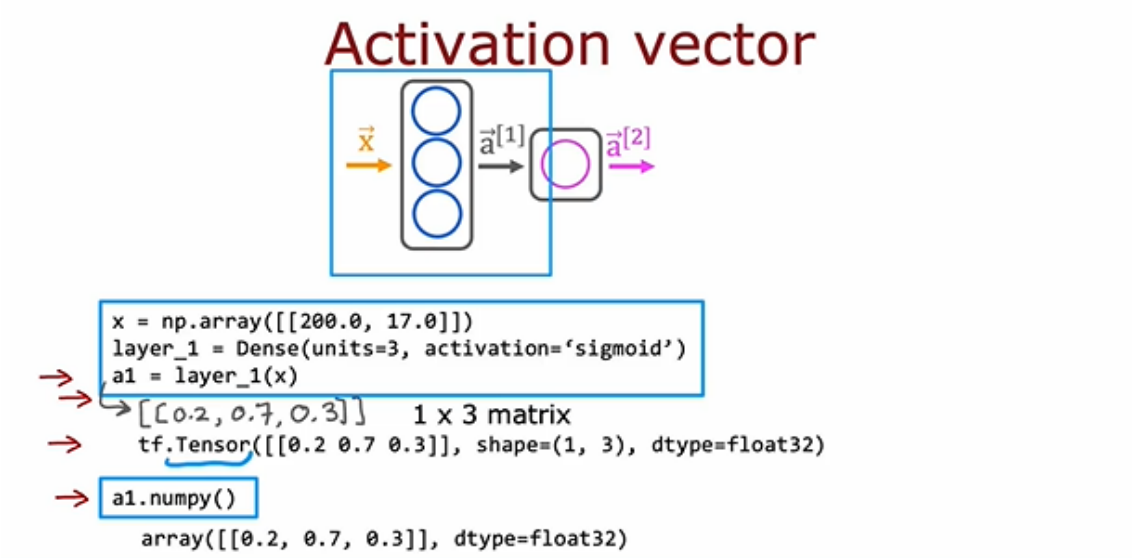




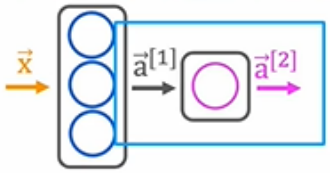
Data in Tensorflow







Activation vector




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

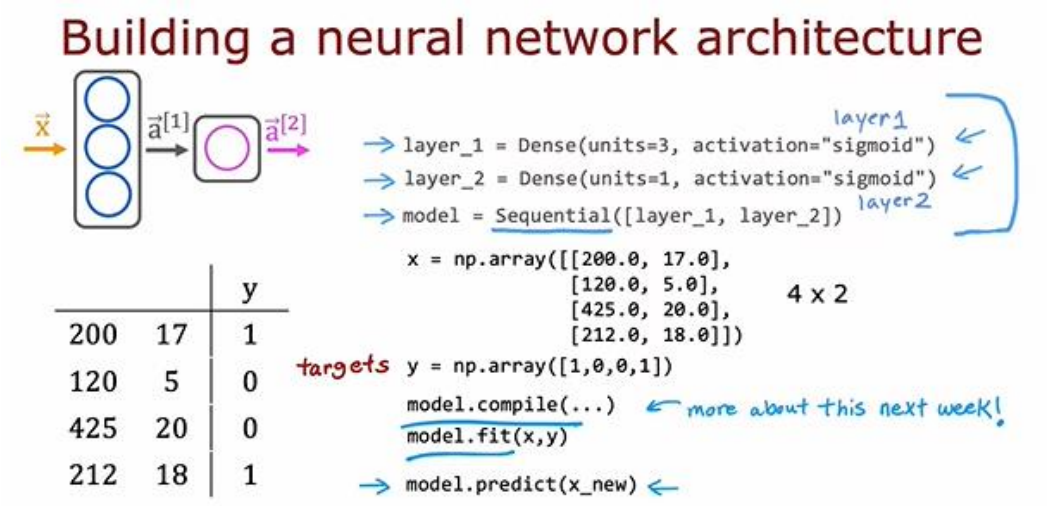
[[0.8]] ← 1 x 1

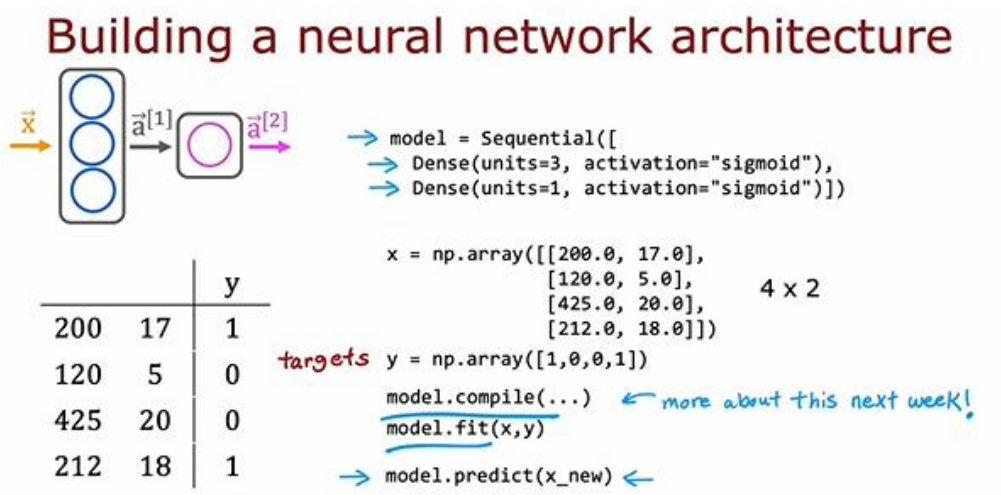
```
tf.Tensor([[0.8]], shape=(1, 1), dtype=float32)
```

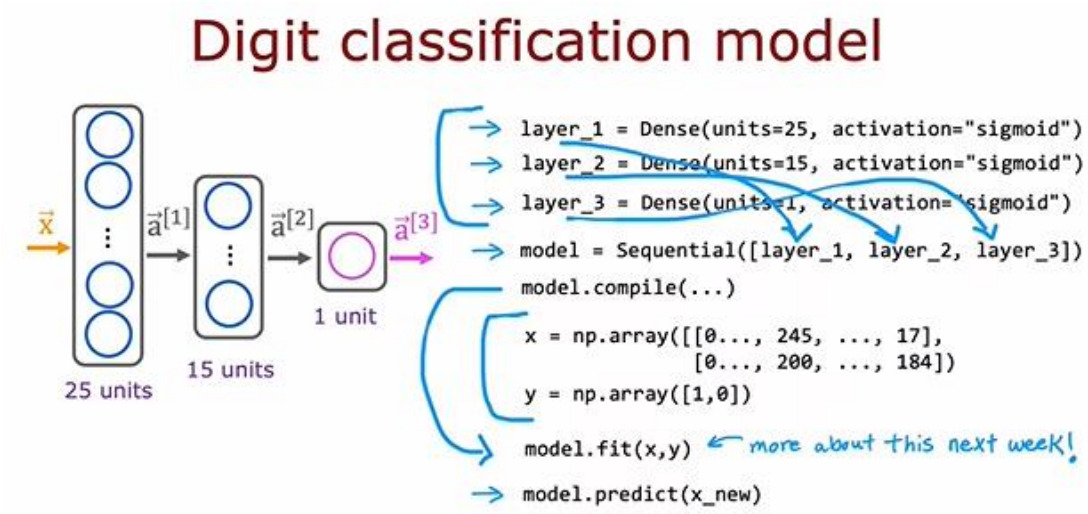
```
a2.numpy()
```

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

Building a Neural Network







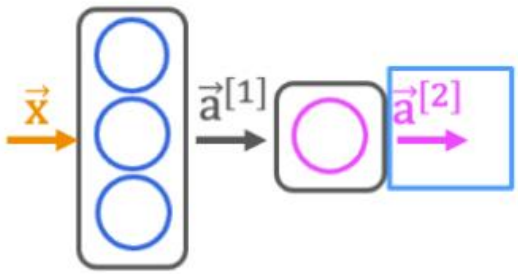
1. For the the following code: 1 point

```
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



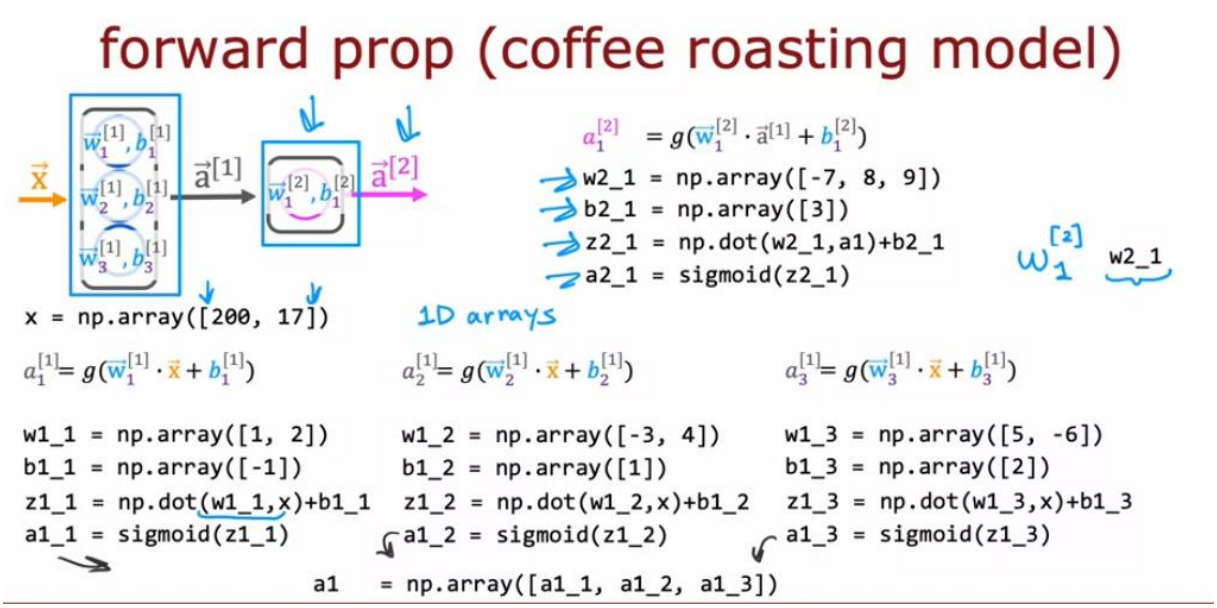

```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation='sigmoid')
a1 = layer_1(x)
```

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

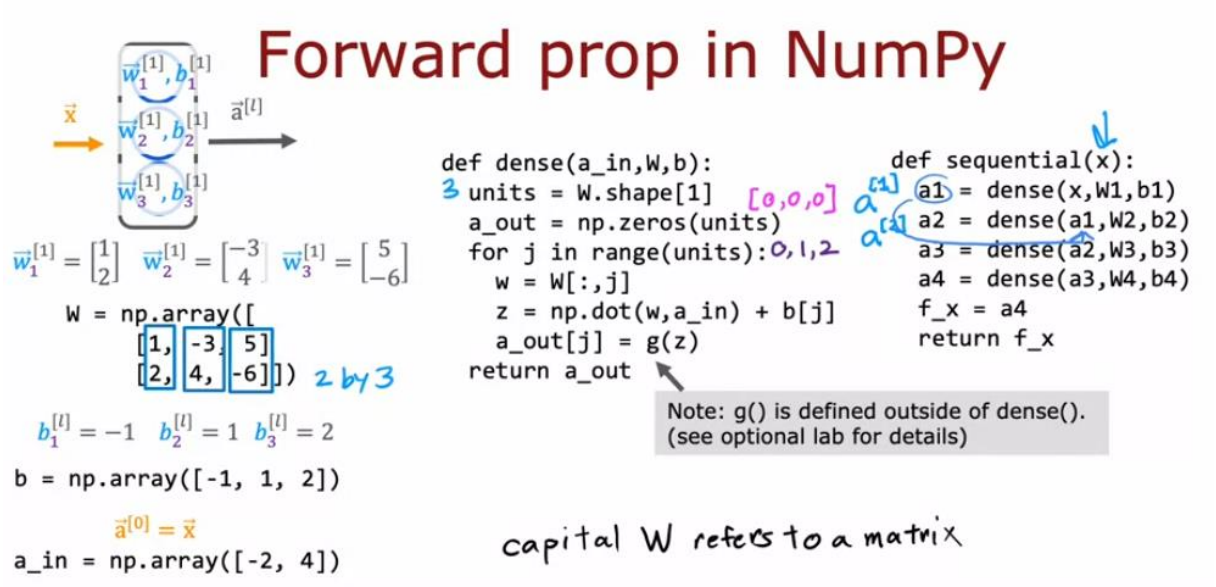
How do you define the second layer of a neural network that has 4 neurons and a sigmoid activation?

- ○ Dense(units=[4], activation=['sigmoid'])
- ○ Dense(units=4)
- ◉ Dense(units=4, activation='sigmoid')
- ○ Dense(layer=2, units=4, activation = 'sigmoid')

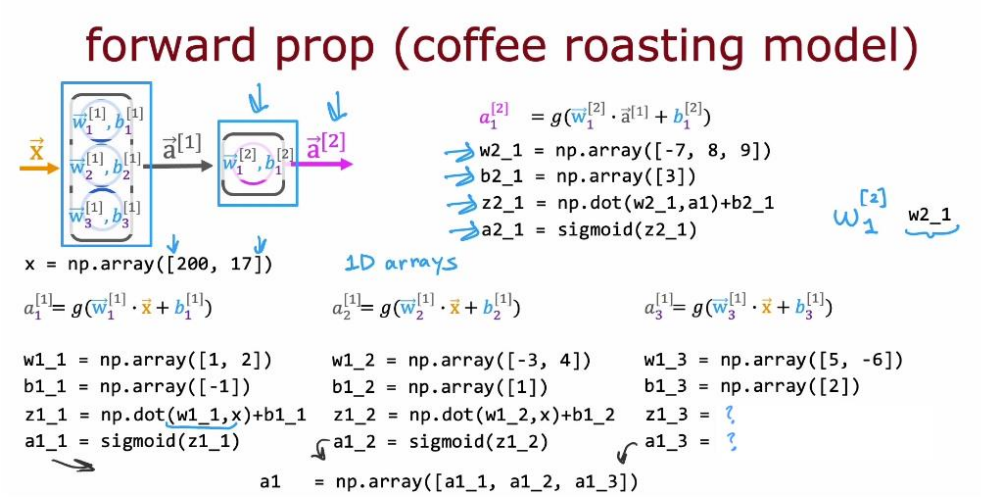
Forward prop in a single layer



Genearal implemantation of forward propagation



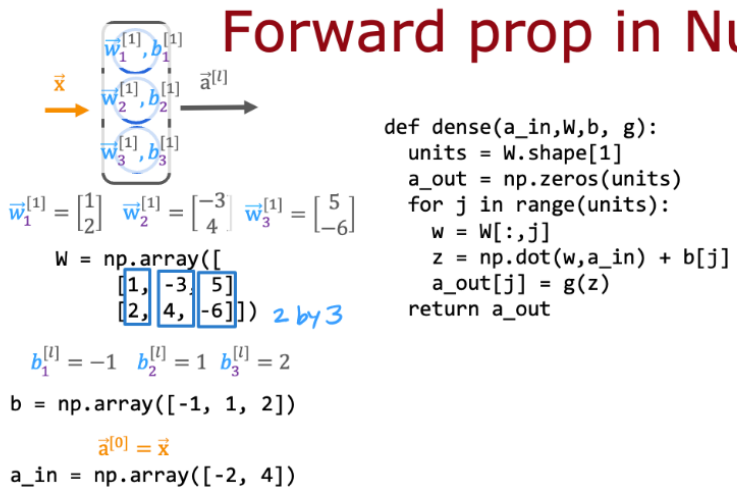
1. 1 point



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)
- **(selected)** z1_3 = np.dot(w1_3, x) + b1_3
  a1_3 = sigmoid(z1_3)

2. 1 point

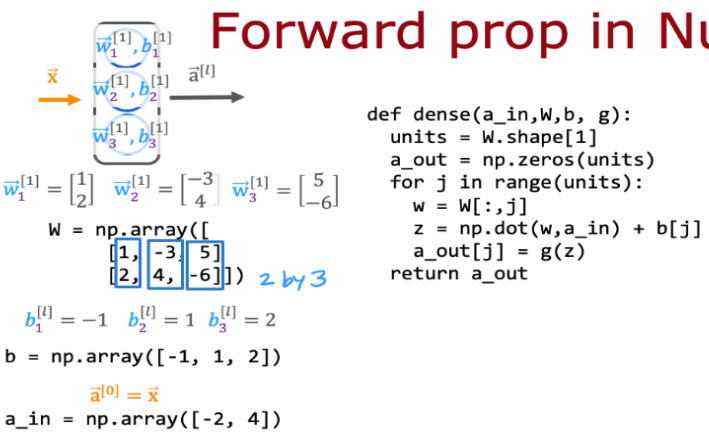


According to the lecture, when coding up the numpy array W, where would you place the w parameters for each neuron?

- In the rows of W.
- **(selected)** In the columns of W.

3.

1 point



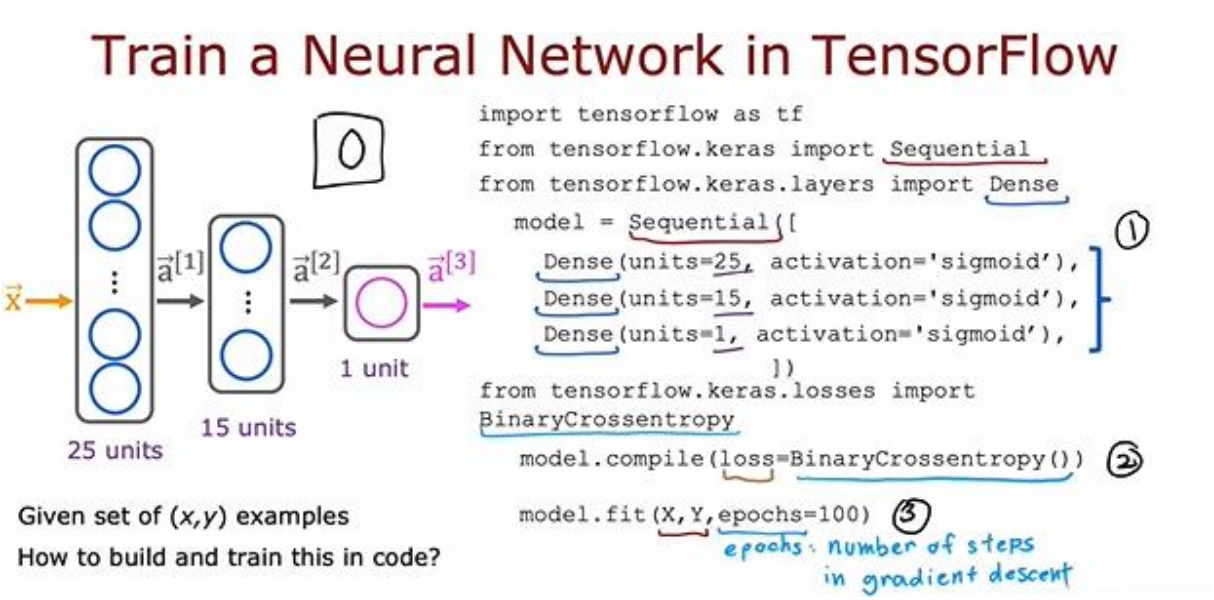
For the code above in the "dense" function that defines a single layer of neurons, how many times does the code go through the "for loop"? Note that W has 2 rows and 3 columns.

- ○ 2 times
- ○ 6 times
- ◉ 3 times
- ○ 5 times

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.

Tensorflow Implementation



Training Details

Model Training Steps

TensorFlow

1. specify how to compute output given input x and parameters w,b (define model)

   $f_{\vec{w},b}(\vec{x}) = ?$

2. specify loss and cost

   $L(f_{\vec{w},b}(\vec{x}), y)$ 1 example

   $J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m} L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)})$

3. Train on data to minimize $J(\vec{w}, b)$

logistic regression

```
z = np.dot(w,x)+ b

f_x = 1/(1+np.exp(-z))
```

logistic loss

```
loss = -y * np.log(f_x)
   -(1-y) * np.log(1-f_x)
```

```
w = w - alpha * dj_dw
b = b - alpha * dj_db
```

neural network

```
model = Sequential([
          Dense(...)
          Dense(...)
          Dense(...)
                    ])
```

binary cross entropy

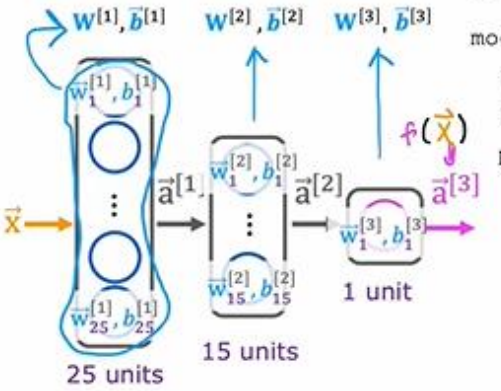
```
model.compile(
loss=BinaryCrossentropy())
```

```
model.fit(X,y,epochs=100)
```

1. Create the model

define the model

$f(\vec{x}) = ?$




```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

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

2. Loss and cost functions

handwritten digit classification problem → binary classification

$$L(f(\vec{x}), y) = -y\log(f(\vec{x})) - (1-y)\log(1-f(\vec{x}))$$

Compare prediction vs. target

logistic loss

also known as binary cross entropy

$$J(\mathbf{W}, \mathbf{B}) = \frac{1}{m}\sum_{i=1}^{m} L(f(\vec{x}^{(i)}), y^{(i)})$$

$\mathbf{W}^{[1]}, \mathbf{W}^{[2]}, \mathbf{W}^{[3]}$ $\vec{b}^{[1]}, \vec{b}^{[2]}, \vec{b}^{[3]}$

$f_{W,B}(\vec{x})$

```
model.compile(loss= BinaryCrossentropy())
```

```
from tensorflow.keras.losses import
  BinaryCrossentropy
```

K Keras

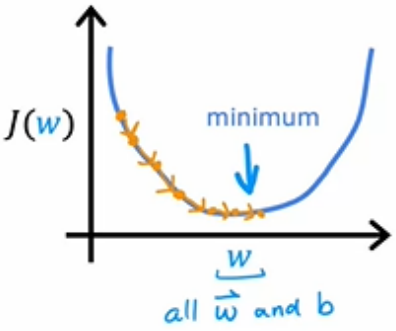
regression (predicting numbers and not categories)

mean squared error

```
model.compile(loss= MeanSquaredError())
```

```
from tensorflow.keras.losses import
  MeanSquaredError
```

3. Gradient descent



all $\vec{w}$ and b

repeat {

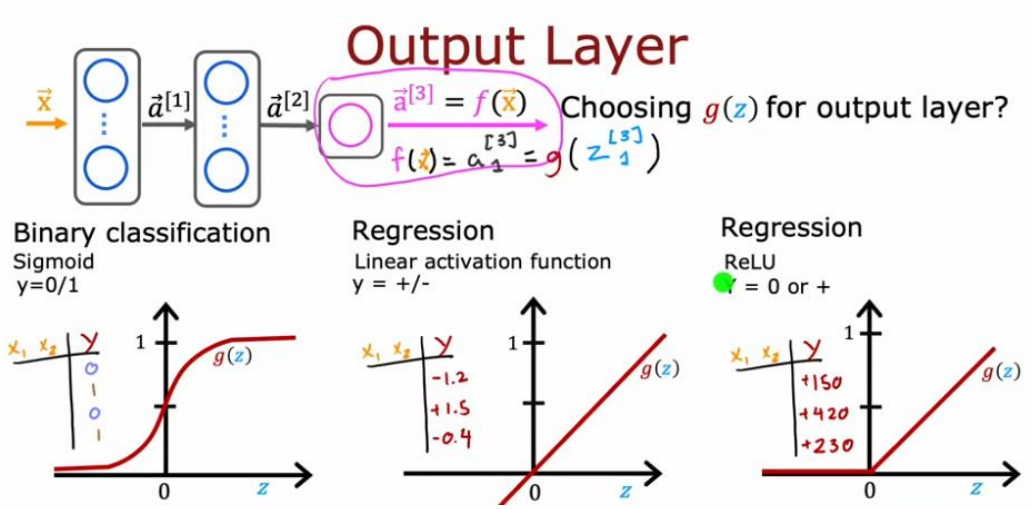
$$w_j^{[l]} = w_j^{[l]} - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$$

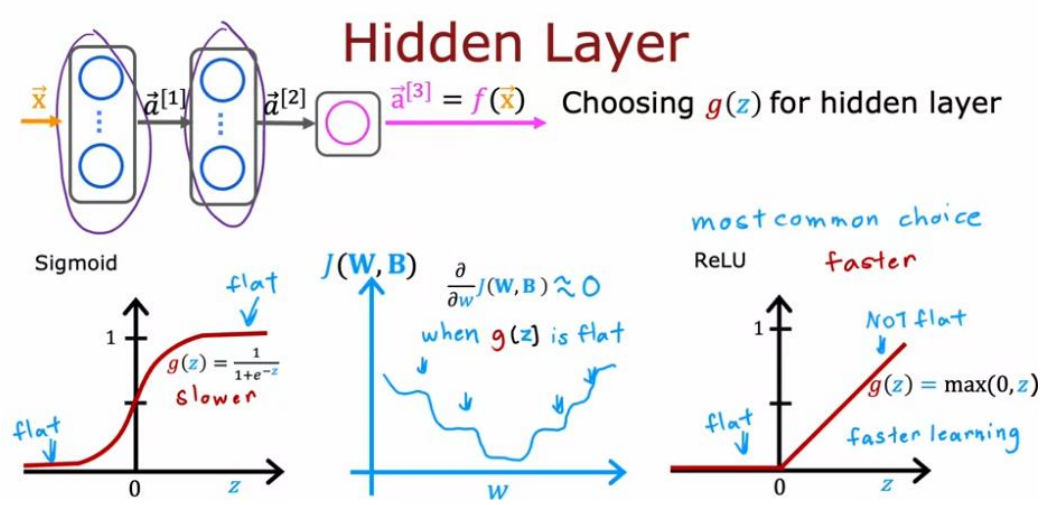
$$b_j^{[l]} = b_j^{[l]} - \alpha \frac{\partial}{\partial bj} J(\vec{w}, b)$$

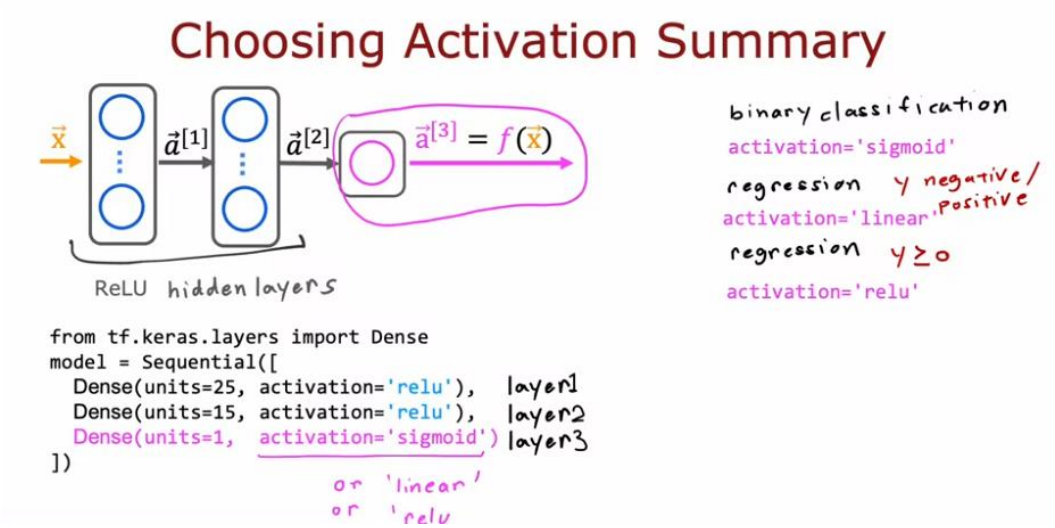
} Compute derivatives for gradient descent using "back propagation"

```
model.fit(X,y,epochs=100)
```

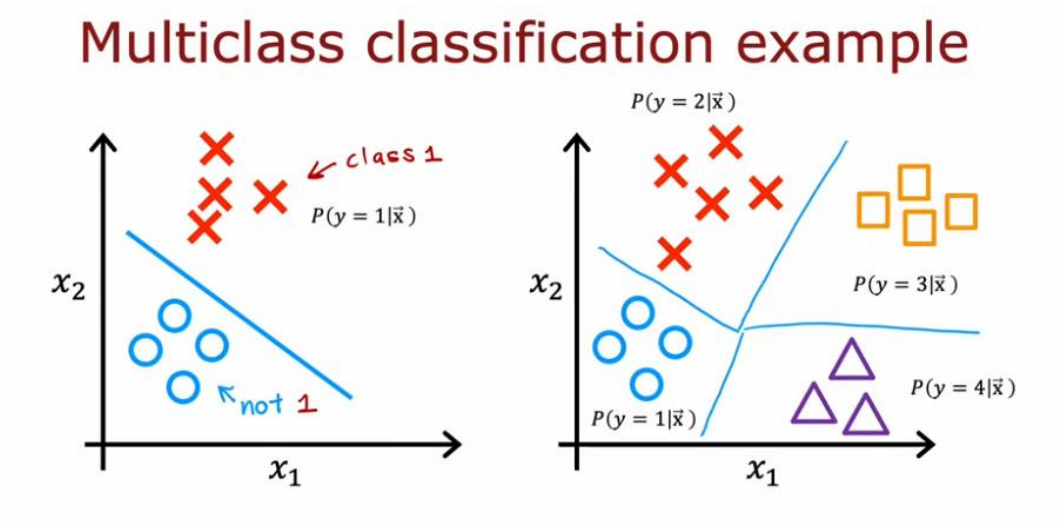
Choosing activation function







Multiclass



Softmax

Logistic regression
(2 possible output values)

$z = \vec{w} \cdot \vec{x} + b$

$a_1 = g(z) = \frac{1}{1+e^{-z}} = P(y=1|\vec{x})$ 0.71

$a_2 = 1 - a_1 = P(y=0|\vec{x})$ 0.29

Softmax regression
(N possible outputs) $y = 1, 2, 3, \ldots, N$

$z_j = \vec{w}_j \cdot \vec{x} + b_j \quad j = 1, \ldots, N$

parameters $w_1, w_2, \ldots, w_N$ $b_1, b_2, \ldots, b_N$

$a_j = \frac{e^{z_j}}{\sum_{k=1}^{N} e^{z_k}} = P(y=j|\vec{x})$

note: $a_1 + a_2 + \ldots + a_N = 1$

Softmax regression (4 possible outputs) $y = 1, 2, 3, 4$

$z_1 = \vec{w}_1 \cdot \vec{x} + b_1$

$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=1|\vec{x})$ 0.30

$z_2 = \vec{w}_2 \cdot \vec{x} + b_2$

$a_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=2|\vec{x})$ 0.20

$z_3 = \vec{w}_3 \cdot \vec{x} + b_3$

$a_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=3|\vec{x})$ 0.15

$z_4 = \vec{w}_4 \cdot \vec{x} + b_4$

$a_4 = \frac{e^{z_4}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=4|\vec{x})$ 0.35

Cost

Logistic regression

$z = \vec{w} \cdot \vec{x} + b$

$a_1 = g(z) = \frac{1}{1+e^{-z}} = P(y=1|\vec{x})$

$a_2 = 1 - a_1 = P(y=0|\vec{x})$

$loss = -y \log a_1 - (1-y) \log(1-a_1)$

if $y=1$ if $y=0$

$J(\vec{w}, b) =$ average loss

Softmax regression

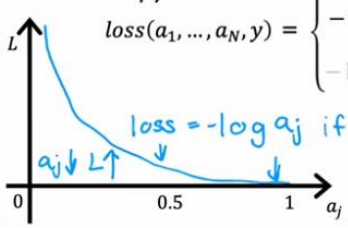
$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + \cdots + e^{z_N}} = P(y=1|\vec{x})$

$\vdots$

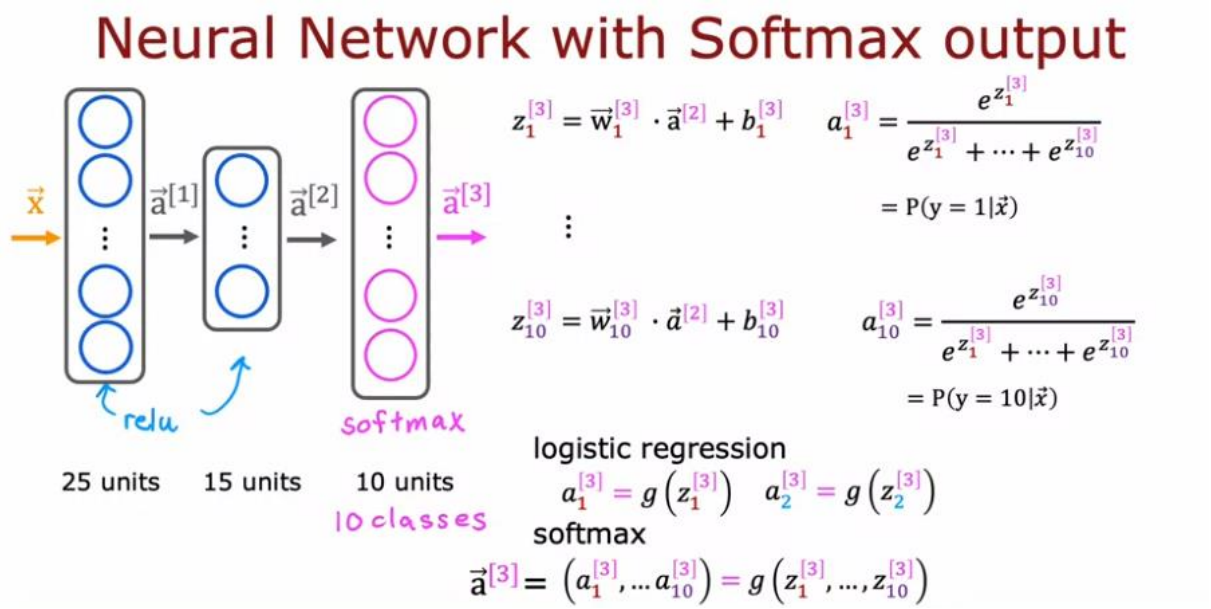
$a_N = \frac{e^{z_N}}{e^{z_1} + e^{z_2} + \cdots + e^{z_N}} = P(y=N|\vec{x})$

Crossentropy loss

$loss(a_1, \ldots, a_N, y) = \begin{cases} -\log a_1 & \text{if } y=1 \\ -\log a_2 & \text{if } y=2 \\ \vdots & \\ -\log a_N & \text{if } y=N \end{cases}$



Neural Network with Softmax output



MNIST with softmax

1. specify the model

$f_{\vec{w},b}(\vec{x})$ =?

2. specify loss and cost

$L(f_{\vec{w},b}(\vec{x}), y)$

3. Train on data to minimize $J(\vec{w}, b)$

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
model = Sequential([
  Dense(units=25, activation='relu'),
  Dense(units=15, activation='relu'),
  Dense(units=10, activation='softmax')
                  ])
from tensorflow.keras.losses import
  SparseCategoricalCrossentropy
model.compile(loss= SparseCategoricalCrossentropy() )
model.fit(X,Y,epochs=100)
```

Note: better (recommended) version later.

Don't use the version shown here!

Improved implementation of softmax

Numerical Roundoff Errors

More numerically accurate implementation of logistic loss:

Logistic regression:

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

Original loss

$$loss = -y\log(a) - (1-y)\log(1-a)$$

More accurate loss (in code)

$$loss = -y\log\left(\frac{1}{1+e^{-z}}\right) - (1-y)\log\left(1-\frac{1}{1+e^{-z}}\right)$$

```
model = Sequential([
  Dense(units=25, activation='relu'),
  Dense(units=15, activation='relu'),
  Dense(units=10, activation='sigmoid')
  ])
model.compile(loss=BinaryCrossEntropy() )
```

```
model.compile(loss=BinaryCrossEntropy(from_logits=True) )
```

More numerically accurate implementation of softmax

Softmax regression

$$(a_1, \ldots, a_{10}) = g(z_1, \ldots, z_{10})$$

$$\text{Loss} = L(\vec{a}, y) = \begin{cases} -\log a_1 & \text{if } y = 1 \\ \vdots & \\ -\log a_{10} & \text{if } y = 10 \end{cases}$$

```
model = Sequential([
  Dense(units=25, activation='relu'),
  Dense(units=15, activation='relu'),
  Dense(units=10, activation='softmax')
  ])
model.compile(loss=SparseCategoricalCrossEntropy() )
```

More Accurate

$$L(\vec{a}, y) = \begin{cases} -\log \dfrac{e^{z_1}}{e^{z_1}+\cdots+e^{z_{10}}} & \text{if } y = 1 \\ \vdots & \\ -\log \dfrac{e^{z_{10}}}{e^{z_1}+\cdots+e^{z_{10}}} & \text{if } y = 10 \end{cases}$$

```
model.compile(loss=SparseCategoricalCrossEntropy(from_logits=True))
```

MNIST (more numerically accurate)

model

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
model = Sequential([
  Dense(units=25, activation='relu'),
  Dense(units=15, activation='relu'),
  Dense(units=10, activation='linear') ])
```

loss

```
from tensorflow.keras.losses import
  SparseCategoricalCrossentropy
model.compile(...,loss=SparseCategoricalCrossentropy(from_logits=True) )
```

fit

```
model.fit(X,Y,epochs=100)
```

predict

```
logits = model(X)
f_x = tf.nn.softmax(logits)
```

not $a_1 \ldots a_{10}$ is $z_1 \ldots$

logistic regression (more numerically accurate)

model

```
model = Sequential([
  Dense(units=25, activation='sigmoid'),
  Dense(units=15, activation='sigmoid'),
  Dense(units=1, activation='linear')
                  ])
from tensorflow.keras.losses import
  BinaryCrossentropy
```

loss

```
model.compile(..., BinaryCrossentropy(from_logits=True)) )
```

fit

```
model.fit(X,Y,epochs=100)
```

predict

```
logit = model(X)
f_x = tf.nn.sigmoid(logit)
```

z

Classification with multiple outputs

Multi-label Classification

Is there a car? yes
Is there a bus? no
Is there a pedestrian yes

$y = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

no
no
yes

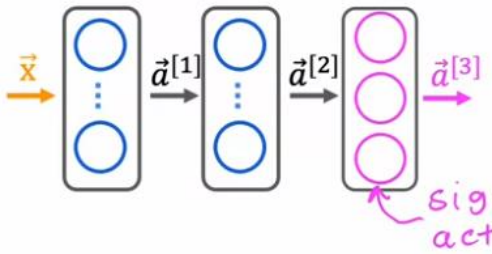
$y = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

yes
yes
no

$y = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$

Multi-label Classification

Alternatively, train one neural network with three outputs



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

car
bus
pedestrian

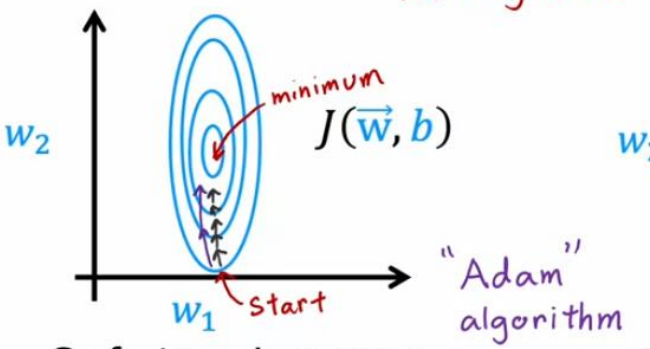
sigmoid activations

Advanced Optimization

Gradient Descent

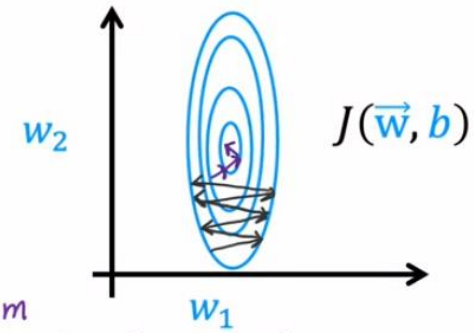
$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$$

α: learning rate



"Adam" algorithm

Go faster – increase α

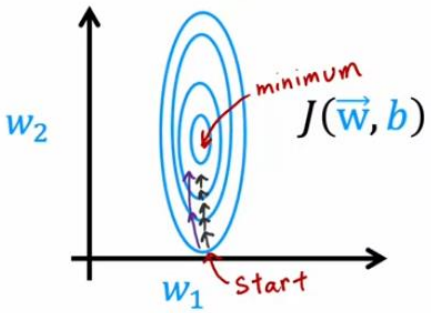


Go slower – decrease α

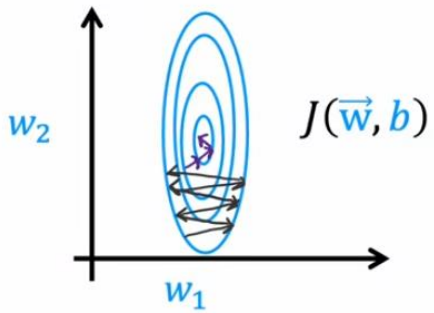
Adam Algorithm Intuition

Adam: Adaptive Moment estimation — not just one α

$$w_1 = w_1 - \alpha_1 \frac{\partial}{\partial w_1} J(\vec{w}, b)$$
$$\vdots$$
$$w_{10} = w_{10} - \alpha_{10} \frac{\partial}{\partial w_{10}} J(\vec{w}, b)$$
$$b = b - \alpha_{11} \frac{\partial}{\partial b} J(\vec{w}, b)$$



If w_j (or b) keeps moving in same direction, increase α_j.



If w_j (or b) keeps oscillating, reduce α_j.

MNIST Adam

model

```
model = Sequential([
        tf.keras.layers.Dense(units=25, activation='sigmoid'),
        tf.keras.layers.Dense(units=15, activation='sigmoid'),
        tf.keras.layers.Dense(units=10, activation='linear')
])
```

compile

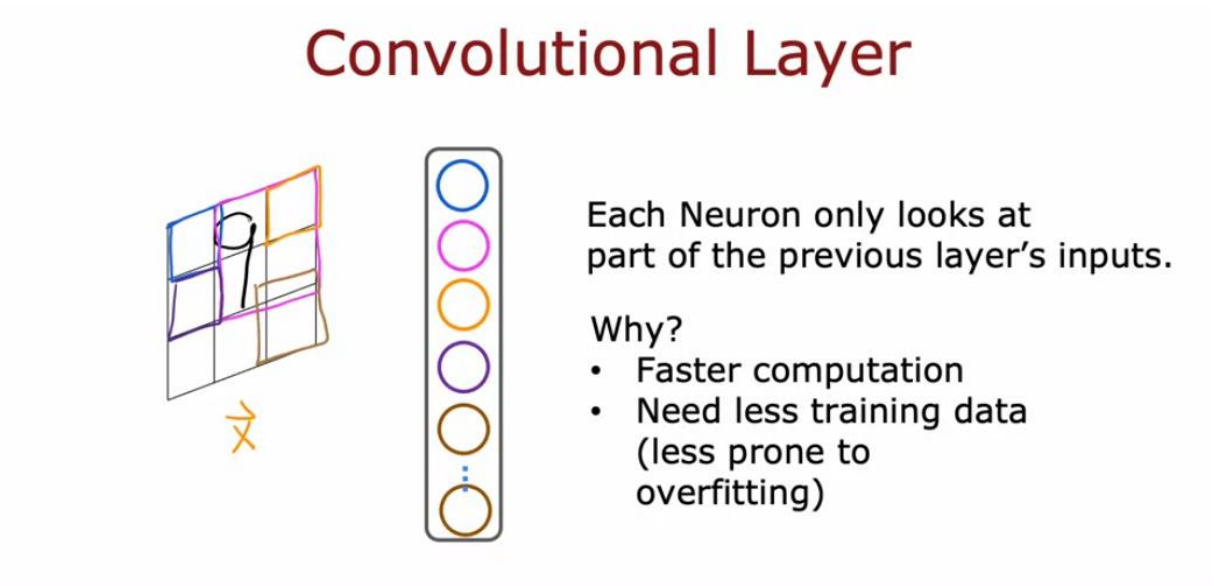
$\alpha = 10^{-3} = 0.001$

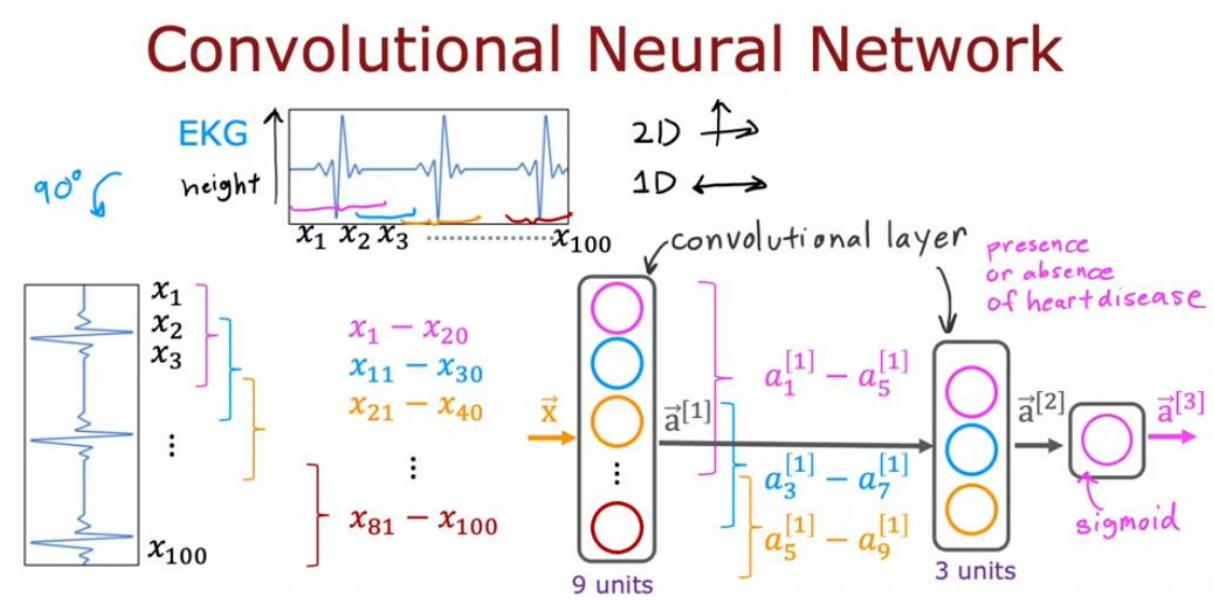
```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True))
```

fit

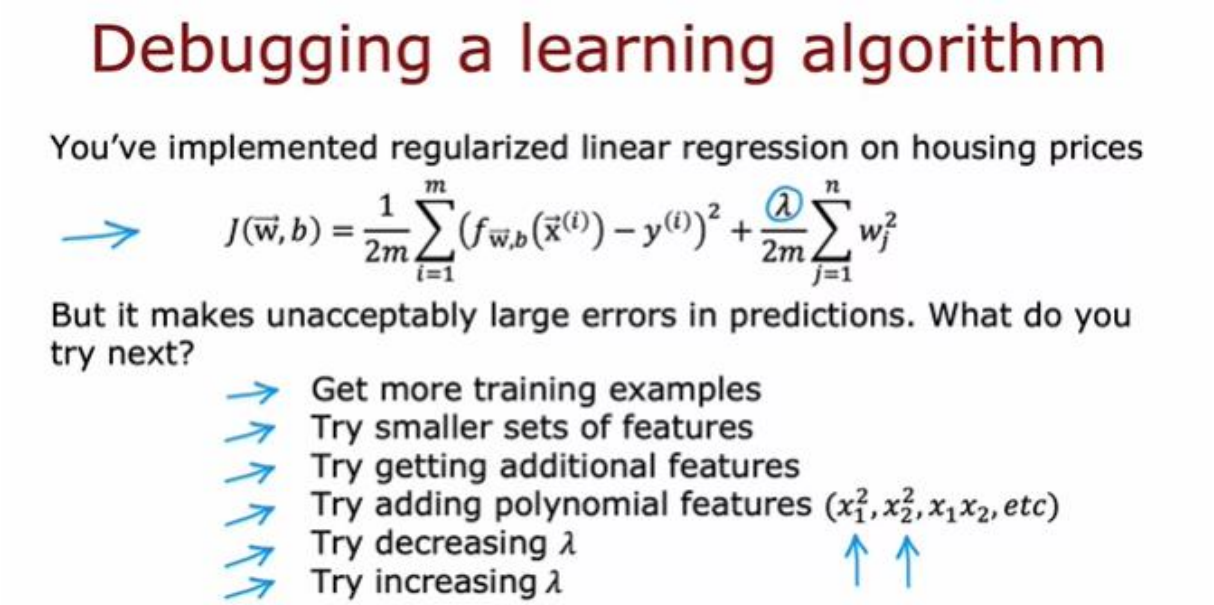
```
model.fit(X,Y,epochs=100)
```

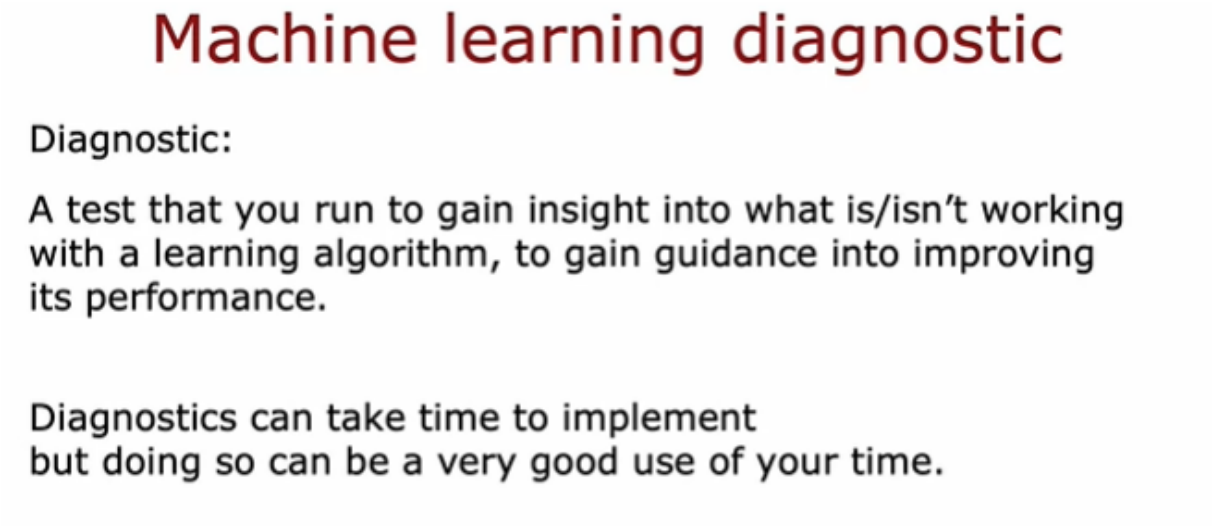
Additional Layer Types





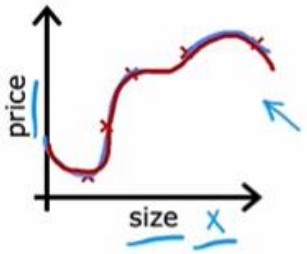
Deciding what to try next





Evaluating a model

Evaluating your model



Model fits the training data well but will fail to generalize to new examples not in the training set.

$x = x_1$

$f_{\vec{w},b}(\vec{x}) = w_1 x + w_2 x^2 + \cdots + w_n x^n + b$

$w_4 x^4$

just x

- x_1 = size in feet^2
- x_2 = no. of bedrooms
- x_3 = no. of floors
- x_4 = age of home in years

$f(\vec{x})$

Evaluating your model

Dataset:

	size	price
70%	2104	400
	1600	330
	2400	369
	1416	232
	3000	540
	1985	300
	1534	315
30%	1427	199
	1380	212
	1494	243

training set → $(x^{(1)}, y^{(1)})$, $(x^{(2)}, y^{(2)})$, ⋮, $(x^{(m_{train})}, y^{(m_{train})})$

m_{train} = no. training examples = 7

test set → $(x_{test}^{(1)}, y_{test}^{(1)})$, ⋮, $(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

m_{test} = no. test examples = 3

Train/test procedure for linear regression (with squared error cost)

Fit parameters by minimizing cost function $J(\vec{w}, b)$

$$J(\vec{w}, b) = \left[\frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m_{train}} \sum_{j=1}^{n} w_j^2\right]$$

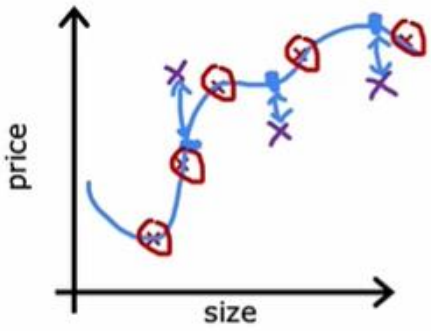
Compute test error:

$$J_{test}(\vec{w}, b) = \frac{1}{2m_{test}} \left[\sum_{i=1}^{m_{test}} \left(f_{\vec{w},b}(\vec{x}_{test}^{(i)}) - y_{test}^{(i)}\right)^2\right] \quad \sum_{j=1}^{n} w_j^2$$

Compute training error:

$$J_{train}(\vec{w}, b) = \frac{1}{2m_{train}} \left[\sum_{i=1}^{m_{train}} \left(f_{\vec{w},b}(\vec{x}_{train}^{(i)}) - y_{train}^{(i)}\right)^2\right]$$

Train/test procedure for linear regression (with squared error cost)



X = train

X = test

$J_{train}(\vec{w}, b)$ will be low

$J_{test}(\vec{w}, b)$ will be high

Train/test procedure for classification problem

0/1

Fit parameters by minimizing $J(\vec{w}, b)$ to find $\vec{w}, b$

E.g.,

$$J(\vec{w}, b) = -\frac{1}{m_{train}} \sum_{i=1}^{m_{train}} \left[y^{(i)} \log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1 - y^{(i)}) \log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right) \right] + \frac{\lambda}{2m_{train}} \sum_{j=1}^{n} w_j^2$$

Compute test error:

$$J_{test}(\vec{w}, b) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} \left[y_{test}^{(i)} \log\left(f_{\vec{w},b}\left(\vec{x}_{test}^{(i)}\right)\right) + \left(1 - y_{test}^{(i)}\right) \log\left(1 - f_{\vec{w},b}\left(\vec{x}_{test}^{(i)}\right)\right) \right]$$

Compute train error:

$$J_{train}(\vec{w}, b) = -\frac{1}{m_{train}} \sum_{i=1}^{m_{train}} \left[y_{train}^{(i)} \log\left(f_{\vec{w},b}\left(\vec{x}_{train}^{(i)}\right)\right) + \left(1 - y_{train}^{(i)}\right) \log\left(1 - f_{\vec{w},b}\left(\vec{x}_{train}^{(i)}\right)\right) \right]$$

Train/test procedure for classification problem

fraction of the test set and the fraction of the train set that the algorithm has misclassified.

$$\hat{y} = \begin{cases} 1 \text{ if } f_{\vec{w},b}(\vec{x}^{(i)}) \geq 0.5 \\ 0 \text{ if } f_{\vec{w},b}(\vec{x}^{(i)}) < 0.5 \end{cases}$$

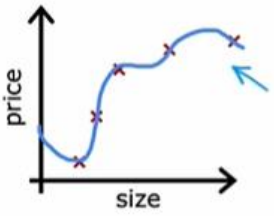
count $\hat{y} \neq y$

$J_{test}(\vec{w}, b)$ is the fraction of the test set that has been misclassified.

$J_{train}(\vec{w}, b)$ is the fraction of the train set that has been misclassified.

Model selection and training/cross validation/test sets

Model selection (choosing a model)



Once parameters $\vec{w}, b$ are fit to the training set, the training error $J_{train}(\vec{w}, b)$ is likely lower than the actual generalization error.

$J_{test}(\vec{w}, b)$ is better estimate of how well the model will generalize to new data compared to $J_{train}(\vec{w}, b)$.

$x = x_1$

$f_{\vec{w},b}(\vec{x}) = w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$

Model selection (choosing a model)

d=1 1. $f_{\vec{w},b}(\vec{x}) = w_1 x + b$ → $w^{<1>}, b^{<1>} \rightarrow J_{test}(w^{<1>}, b^{<1>})$

d=2 2. $f_{\vec{w},b}(\vec{x}) = w_1 x + w_2 x^2 + b$ → $w^{<2>}, b^{<2>} \rightarrow J_{test}(w^{<2>}, b^{<2>})$

d=3 3. $f_{\vec{w},b}(\vec{x}) = w_1 x + w_2 x^2 + w_3 x^3 + b$ → $w^{<3>}, b^{<3>} \rightarrow J_{test}(w^{<3>}, b^{<3>})$

⋮

d=10 10. $f_{\vec{w},b}(\vec{x}) = w_1 x + w_2 x^2 + \cdots + w_{10} x^{10} + b$ → $J_{test}(w^{<10>}, b^{<10>})$

Choose $w_1 x + \cdots + w_5 x^5 + b$ d=5 $J_{test}(w^{<5>}, b^{<5>})$

How well does the model perform? Report test set error $J_{test}(w^{<5>}, b^{<5>})$?

The problem: $J_{test}(w^{<5>}, b^{<5>})$ is likely to be an optimistic estimate of generalization error (ie. $J_{test}(w^{<5>}, b^{<5>})$ < generalization error). Because an extra parameter d (degree of polynomial) was chosen using the test set.

w, b are overly optimistic estimate of generalization error on training data.

Training/cross validation/test set

validation set
development set
dev set

size	price
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243

training set 60% → $(x^{(1)}, y^{(1)}) \ \vdots \ (x^{(m_{train})}, y^{(m_{train})})$ $m_{train} = 6$

cross validation 20% → $(x_{cv}^{(1)}, y_{cv}^{(1)}) \ \vdots \ (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$ $m_{cv} = 2$

test set 20% → $(x_{test}^{(1)}, y_{test}^{(1)}) \ \vdots \ (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$ $m_{test} = 2$

Training/cross validation/test set

Training error: $J_{train}(\vec{w}, b) = \frac{1}{2m_{train}}\left[\sum_{i=1}^{m_{train}} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2\right]$

Cross validation error: $J_{cv}(\vec{w}, b) = \frac{1}{2m_{cv}}\left[\sum_{i=1}^{m_{cv}} \left(f_{\vec{w},b}\left(\vec{x}_{cv}^{(i)}\right) - y_{cv}^{(i)}\right)^2\right]$ (validation error, dev error)

Test error: $J_{test}(\vec{w}, b) = \frac{1}{2m_{test}}\left[\sum_{i=1}^{m_{test}} \left(f_{\vec{w},b}\left(\vec{x}_{test}^{(i)}\right) - y_{test}^{(i)}\right)^2\right]$

Model selection

$d=1$ 1. $f_{\vec{w},b}(\vec{x}) = w_1 x + b$ $w^{\langle 1\rangle}, b^{\langle 1\rangle}$ → $J_{cv}(w^{\langle 1\rangle}, b^{\langle 1\rangle})$

$d=2$ 2. $f_{\vec{w},b}(\vec{x}) = w_1 x + w_2 x^2 + b$ → $J_{cv}(w^{\langle 2\rangle}, b^{\langle 2\rangle})$

$d=3$ 3. $f_{\vec{w},b}(\vec{x}) = w_1 x + w_2 x^2 + w_3 x^3 + b$

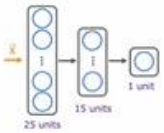
⋮ ⋮ ⋮

$d=10$ 10. $f_{\vec{w},b}(\vec{x}) = w_1 x + w_2 x^2 + \cdots + w_{10} x^{10} + b$ $J_{cv}(w^{\langle 10\rangle}, b^{\langle 10\rangle})$

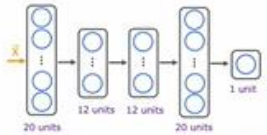
→ Pick $w_1 x + \cdots + w_4 x^4 + b$ $(J_{cv}(w^{<4>}, b^{<4>}))$

Estimate generalization error using test the set: $J_{test}(w^{<4>}, b^{<4>})$

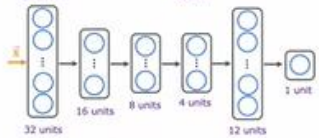
Model selection – choosing a neural network architecture

1. 

$w^{\langle 1\rangle}, b^{\langle 1\rangle}$ $J_{cv}(w^{\langle 1\rangle}, b^{\langle 1\rangle})$

→ 2. 

$w^{\langle 2\rangle}, b^{\langle 2\rangle}$ $J_{cv}(w^{\langle 2\rangle}, b^{\langle 2\rangle})$

3. 

$w^{\langle 3\rangle}, b^{\langle 3\rangle}$ $J_{cv}(w^{\langle 3\rangle}, b^{\langle 3\rangle})$

train, CV

Pick $w^{<2>}, b^{<2>}$

Estimate generalization error using the test set: $J_{test}(w^{<2>}, b^{(<2>)})$

1. 1 point

In the context of machine learning, what is a diagnostic?

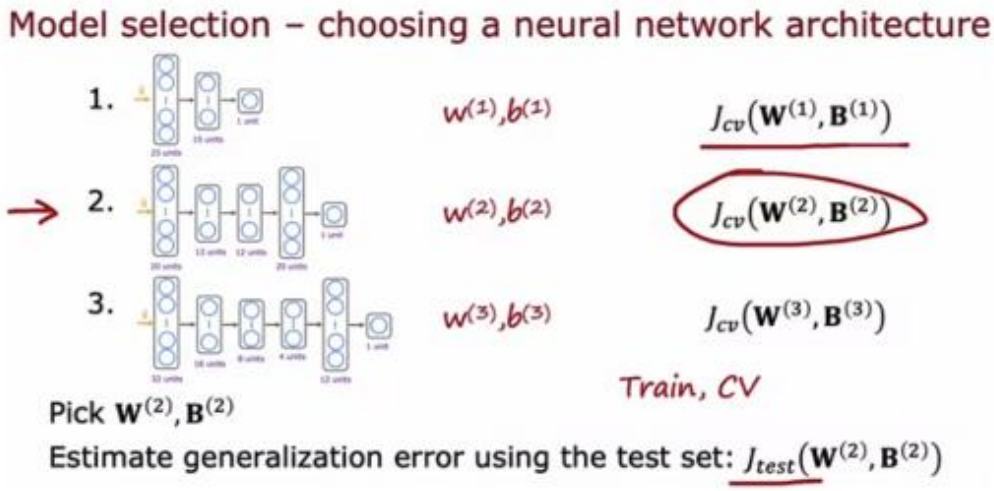
- (●) A test that you run to gain insight into what is/isn't working with a learning algorithm.
- ( ) An application of machine learning to medical applications, with the goal of diagnosing patients' conditions.
- ( ) A process by which we quickly try as many different ways to improve an algorithm as possible, so as to see what works.
- ( ) This refers to the process of measuring how well a learning algorithm does on a test set (data that the algorithm was not trained on).

2. 1 point

True/False? It is always true that the better an algorithm does on the training set, the better it will do on generalizing to new data.

- (●) False
- ( ) True

3. 1 point



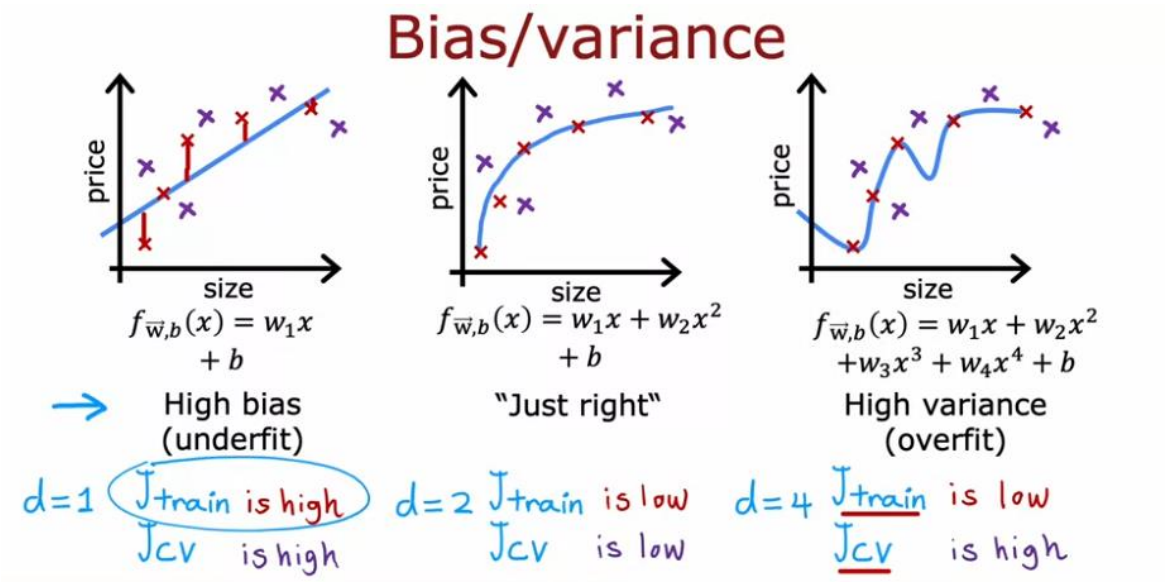
For a classification task; suppose you train three different models using three different neural network architectures. Which data do you use to evaluate the three models in order to choose the best one?

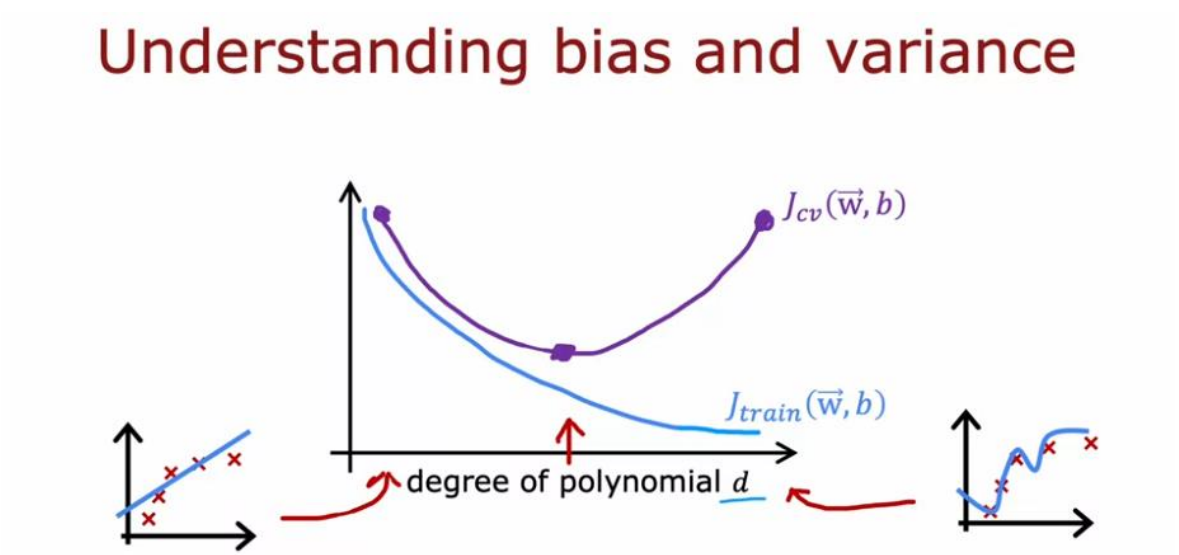
- ( ) The training set
- ( ) The test set
- ( ) All the data -- training, cross validation and test sets put together.
- (●) The cross validation set

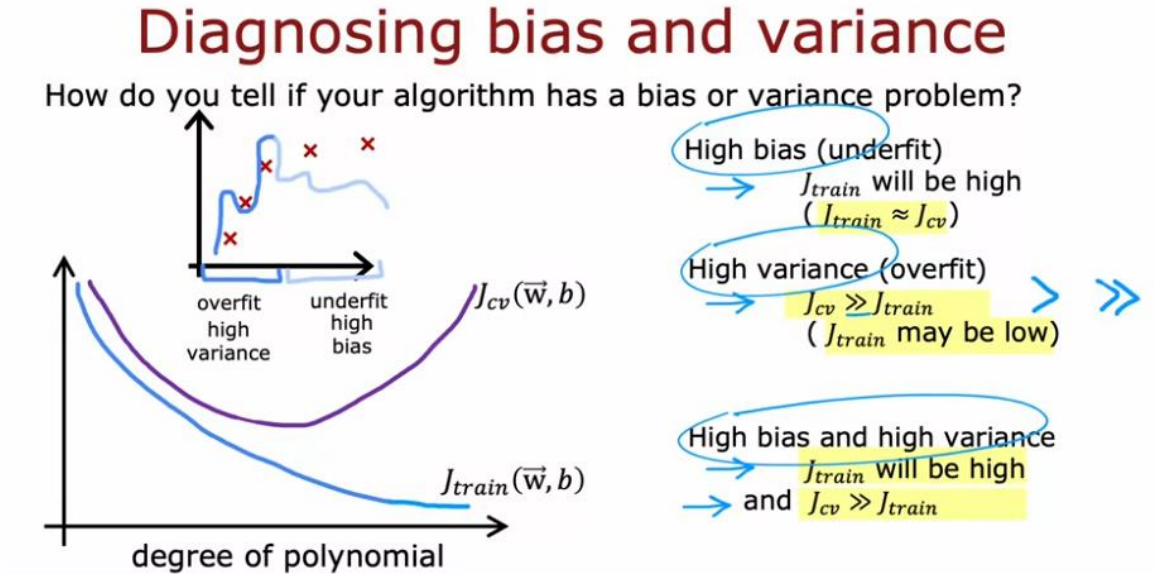
Coursera Honor Code Learn more

☑ 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.

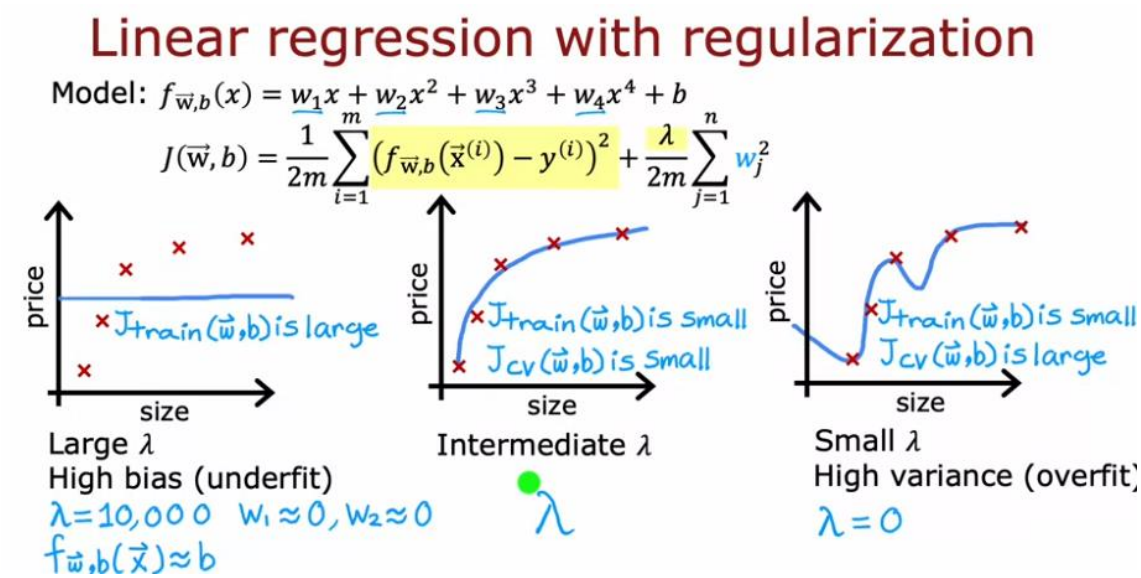
Diagnosing bias and variance

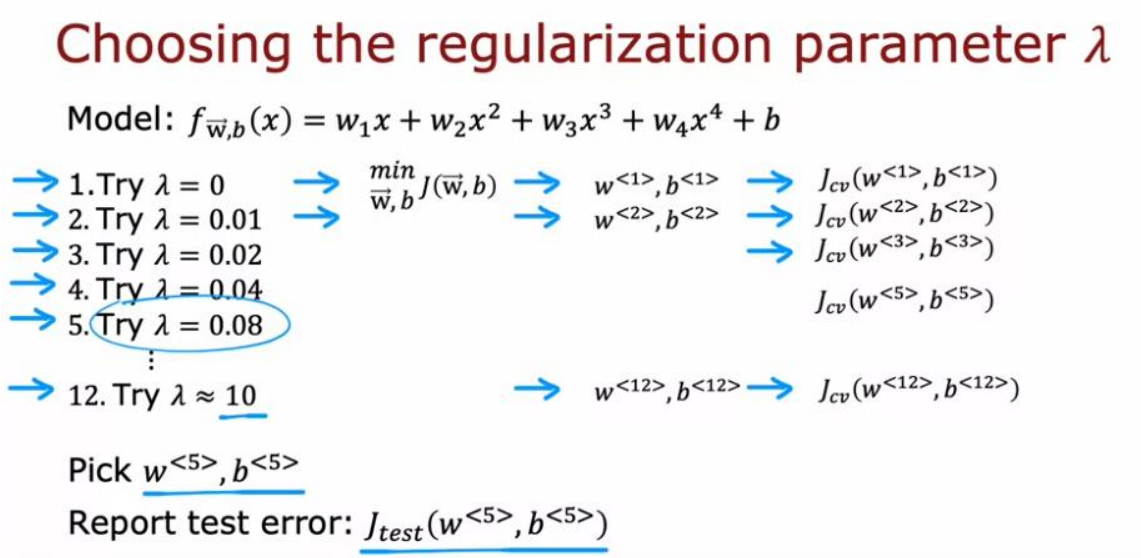


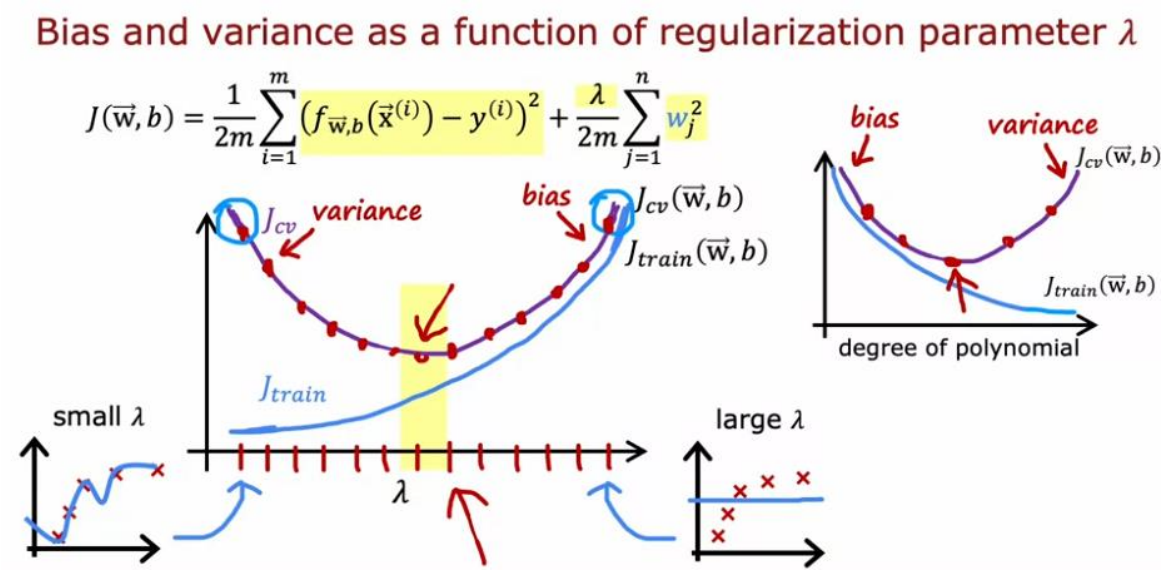




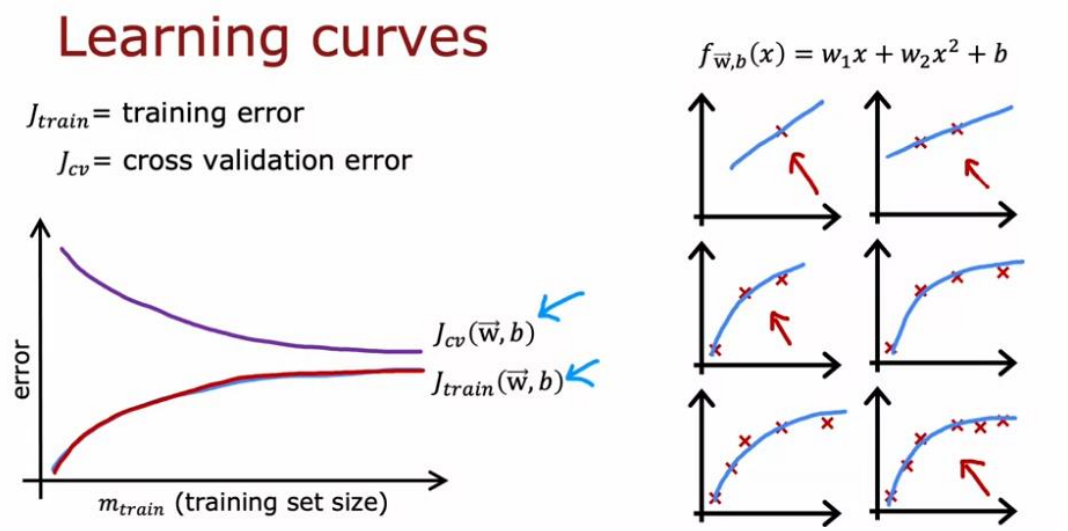
Regularization and bias/variance

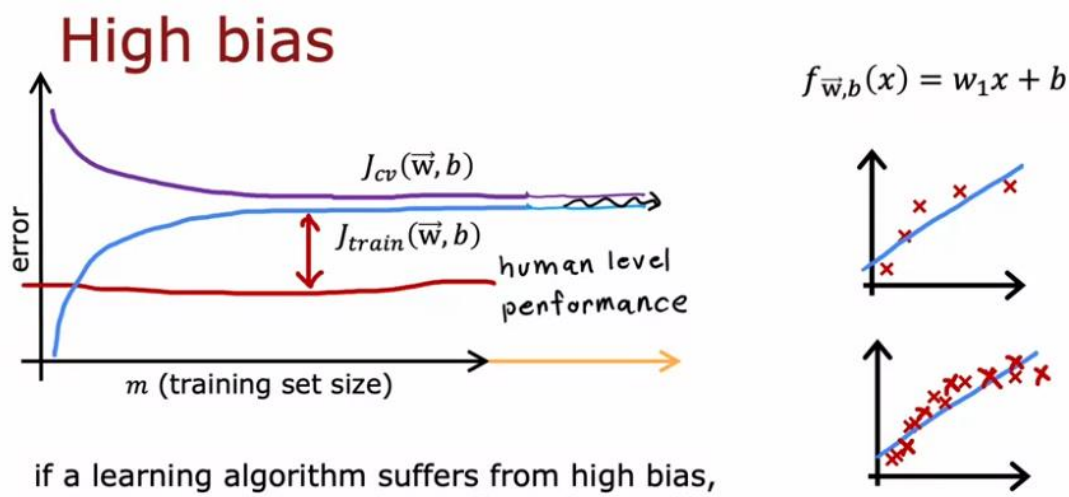




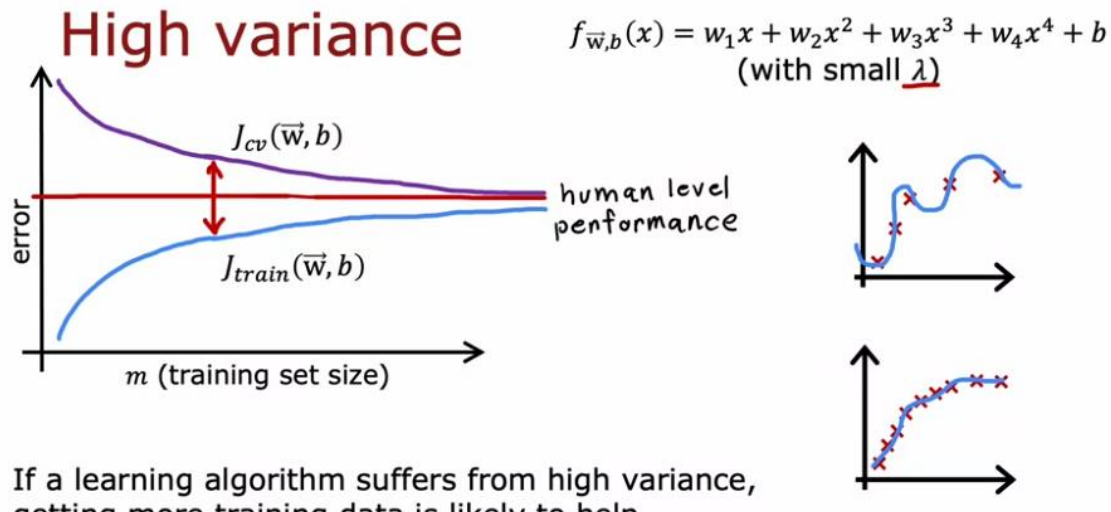


Learning curves





if a learning algorithm suffers from high bias, getting more training data will not (by itself) help much.



If a learning algorithm suffers from high variance, getting more training data is likely to help.

Deciding what to try next revisited

Debugging a learning algorithm

You've implemented regularized linear regression on housing prices

$$J(\vec{w}, b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

But it makes unacceptably large errors in predictions. What do you try next?

→ Get more training examples	fixes high variance
→ Try smaller sets of features $x, x^2, \ldots$	fixes high variance
→ Try getting additional features	fixes high bias
→ Try adding polynomial features $(x_1^2, x_2^2, x_1x_2, etc)$	fixes high bias
→ Try decreasing λ	fixes high bias
→ Try increasing λ	fixes high variance

Bias/variance and neural networks

The bias variance tradeoff

$f_{\vec{w},b}(x) = w_1x + b$

Simple model

High bias

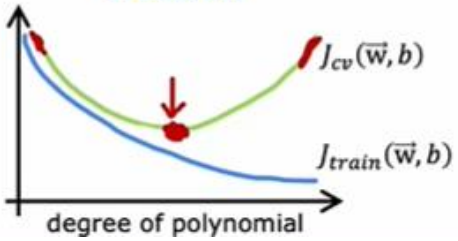
$f_{\vec{w},b}(x) = w_1x + w_2x^2 + b$

$f_{\vec{w},b}(x) = w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$

Complex model

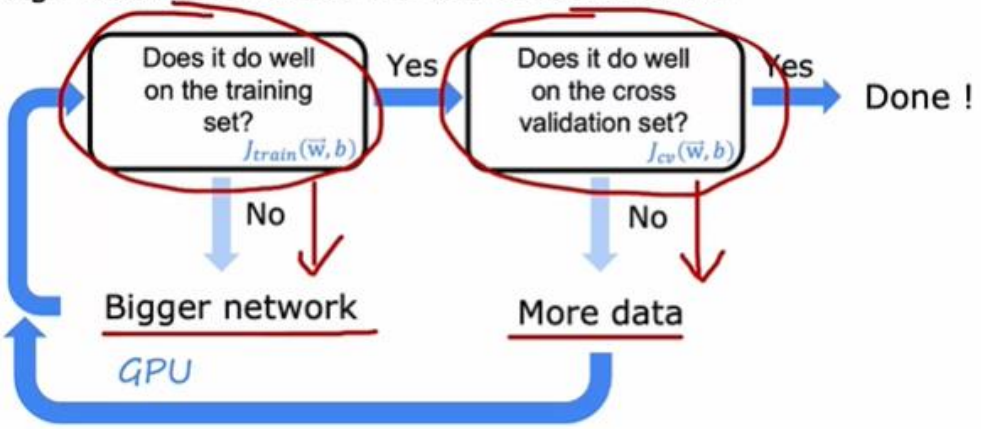
High variance

tradeoff

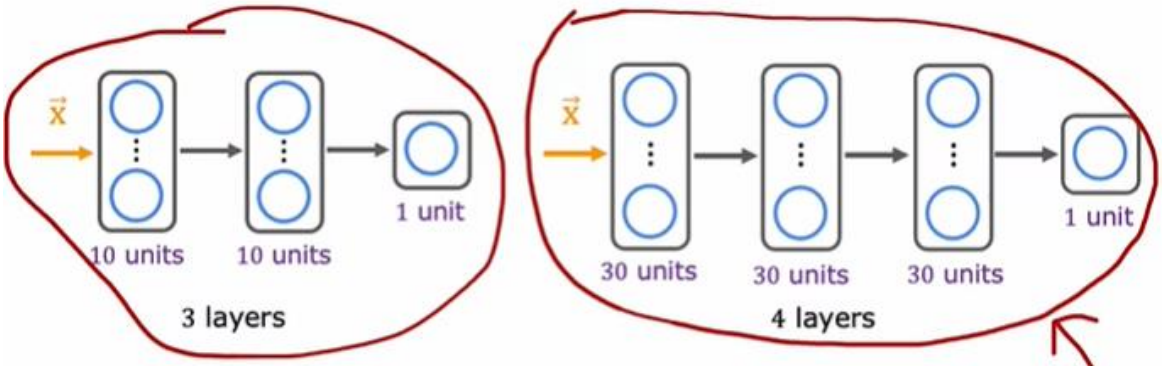


Neural networks and bias variance

Large neural networks are low bias machines



Neural networks and regularization



A large neural network will usually do as well or better than a smaller one so long as regularization is chosen appropriately.

Neural network regularization

$$J(\mathbf{W}, \mathbf{B}) = \frac{1}{m}\sum_{i=1}^{m} L\left(f\left(\vec{\mathbf{x}}^{(i)}\right), y^{(i)}\right) + \frac{\lambda}{2m}\sum_{\text{all weights } \mathbf{W}} \left(w^2\right)$$

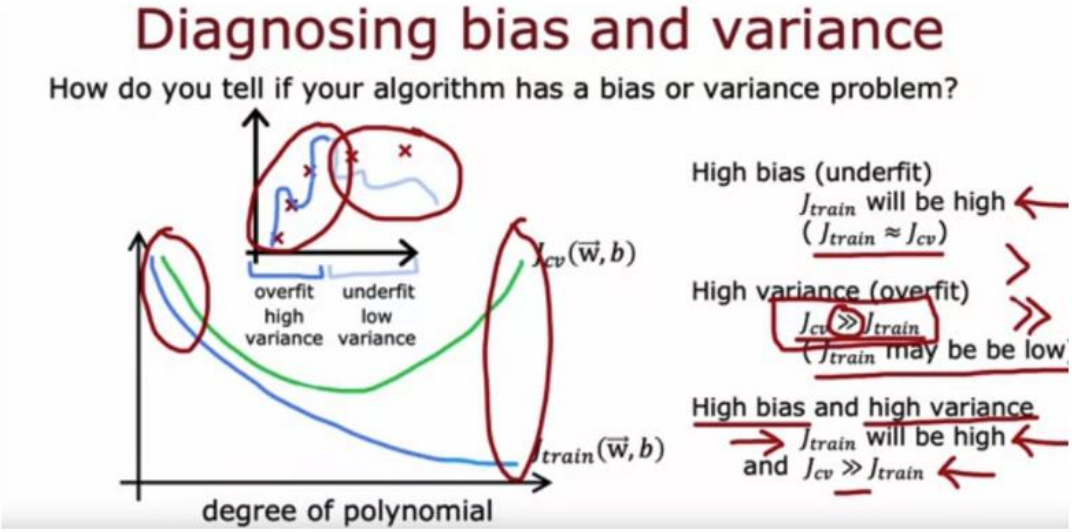
Unregularized MNIST model

```
layer_1 = Dense(units=25, activation="relu")
layer_2 = Dense(units=15, activation="relu")
layer_3 = Dense(units=1, activation="sigmoid")
model = Sequential([layer_1, layer_2, layer_3])
```

Regularized MNIST model

```
layer_1 = Dense(units=25, activation="relu", kernel_regularizer=L2(0.01))
layer_2 = Dense(units=15, activation="relu", kernel_regularizer=L2(0.01))
layer_3 = Dense(units=1, activation="sigmoid", kernel_regularizer=L2(0.01))
model = Sequential([layer_1, layer_2, layer_3])
```

1. 1 / 1 point

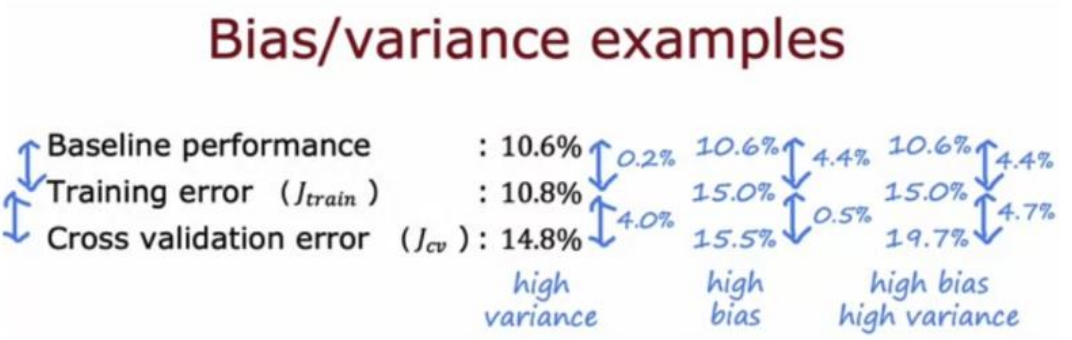


If the model's cross validation error J_{cv} is much higher than the training error J_{train}, this is an indication that the model has...

- (●) high variance
- ( ) Low variance
- ( ) high bias
- ( ) Low bias

Correct
When $J_{cv} >> J_{train}$ (whether J_{train} is also high or not, this is a sign that the model is overfitting to the training data and performing much worse on new examples.

2. 1 / 1 point

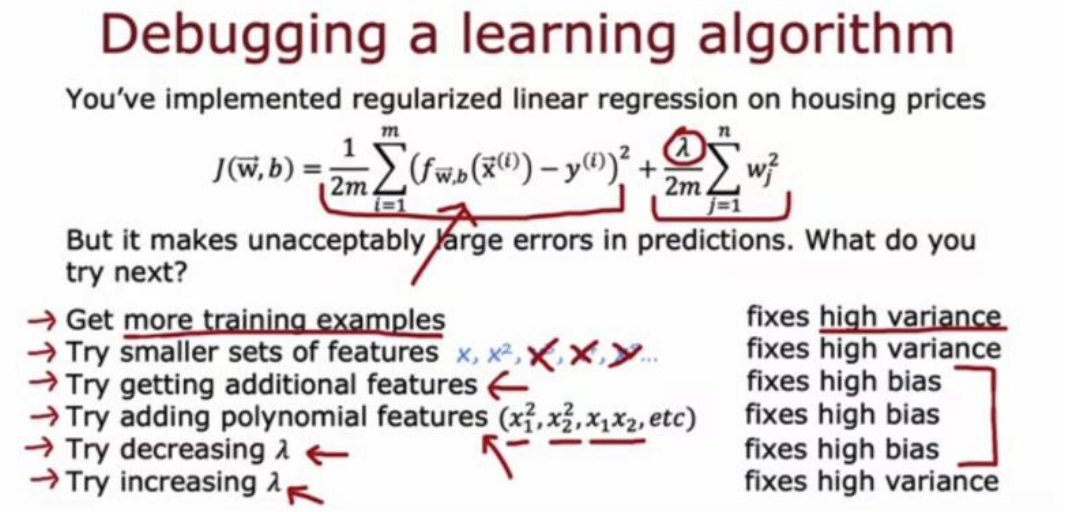


Which of these is the best way to determine whether your model has high bias (has underfit the training data)?

- (●) Compare the training error to the baseline level of performance
- ( ) See if the cross validation error is high compared to the baseline level of performance
- ( ) See if the training error is high (above 15% or so)
- ( ) Compare the training error to the cross validation error.

Correct
Correct. If comparing your model's training error to a baseline level of performance (such as human level performance, or performance of other well-established models), if your model's training error is much higher, then this is a sign that the model has high bias (has underfit).

3. 1 / 1 point



You find that your algorithm has high bias. Which of these seem like good options for improving the algorithm's performance? Hint: two of these are correct.

- [ ] Remove examples from the training set
- [x] Decrease the regularization parameter λ (lambda)

Correct
Correct. Decreasing regularization can help the model better fit the training data.

- [ ] Collect more training examples
- [x] Collect additional features or add polynomial features

Correct
Correct. More features could potentially help the model better fit the training examples.

4. 1 / 1 point

You find that your algorithm has a training error of 2%, and a cross validation error of 20% (much higher than the training error). Based on the conclusion you would draw about whether the algorithm has a high bias or high variance problem, which of these seem like good options for improving the algorithm's performance? Hint: two of these are correct.

- [x] Collect more training data

Correct
Yes, the model appears to have high variance (overfit), and collecting more training examples would help reduce high variance.

- [ ] Reduce the training set size
- [ ] Decrease the regularization parameter λ
- [x] Increase the regularization parameter λ

Correct
Yes, the model appears to have high variance (overfit), and increasing regularization would help reduce high variance.

Iterative loop of ML development