Copyright Notice

DeepLearning.AI

Stanford
ONLINE

Advanced Learning Algorithms

Welcome!

Advanced learning algorithms

Neural Networks 
  inference (prediction)
  training
Practical advice for building machine learning systems 
Decision Trees 

DeepLearning.AI

Stanford
ONLINE

Neural Networks Intuition

Neurons and the brain

Neural networks

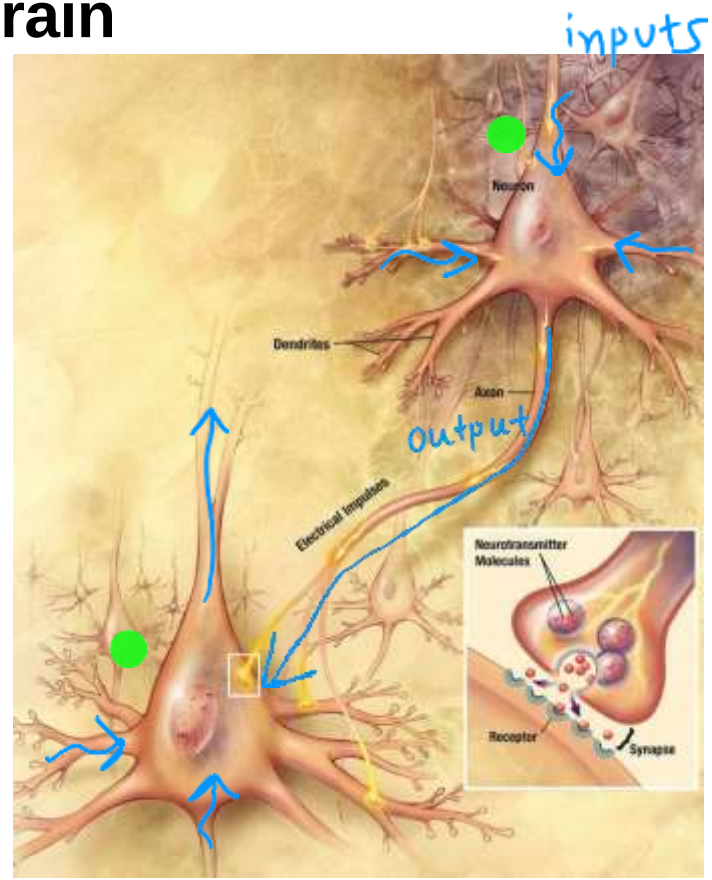
Origins: Algorithms that try to mimic the brain.

Used in the 1980's and early 1990's.
Fell out of favor in the late 1990's.

Resurgence from around 2005.

speech → images → text (NLP) → ...

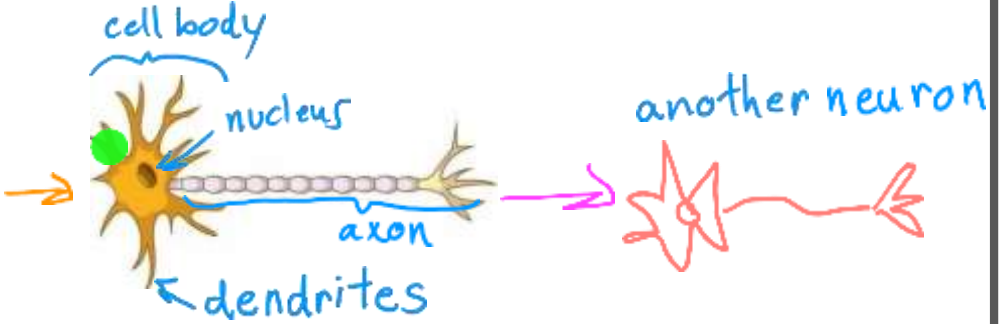
Neurons in the brain



[Credit: US National Institutes of Health, National Institute on Aging]

Biological neuron

inputs outputs



Simplified mathematical model of a neuron

inputs outputs

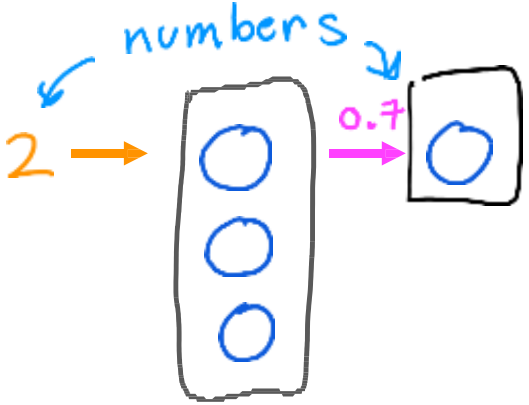


image source: https://biologydictionary.net/sensory-neuron/

Why Now?


Faster computer processors
GPUs
large neural network
medium neural network
small neural network
traditional AI
performance
amount of data
Big Data

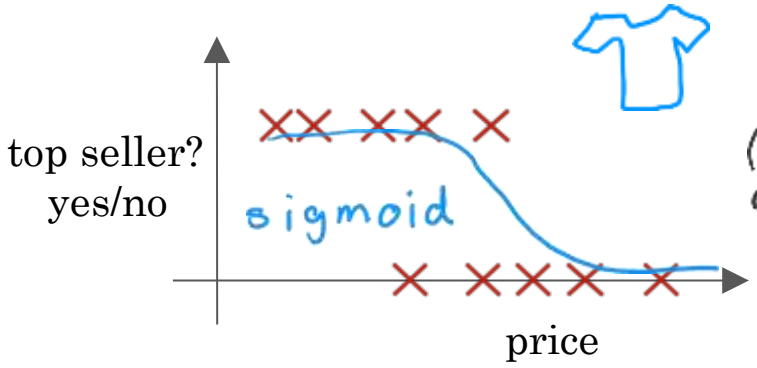
DeepLearning.AI

Stanford
ONLINE

Neural Network Intuition

Demand Prediction

Demand Prediction



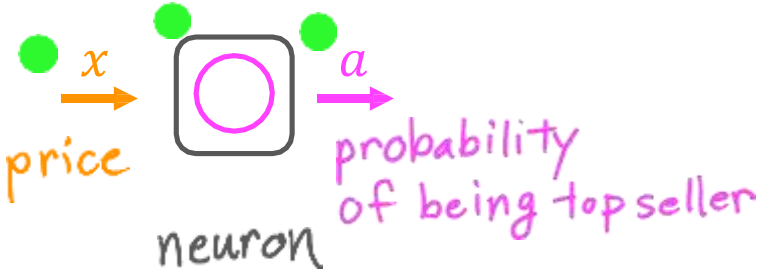
$x = price$

input

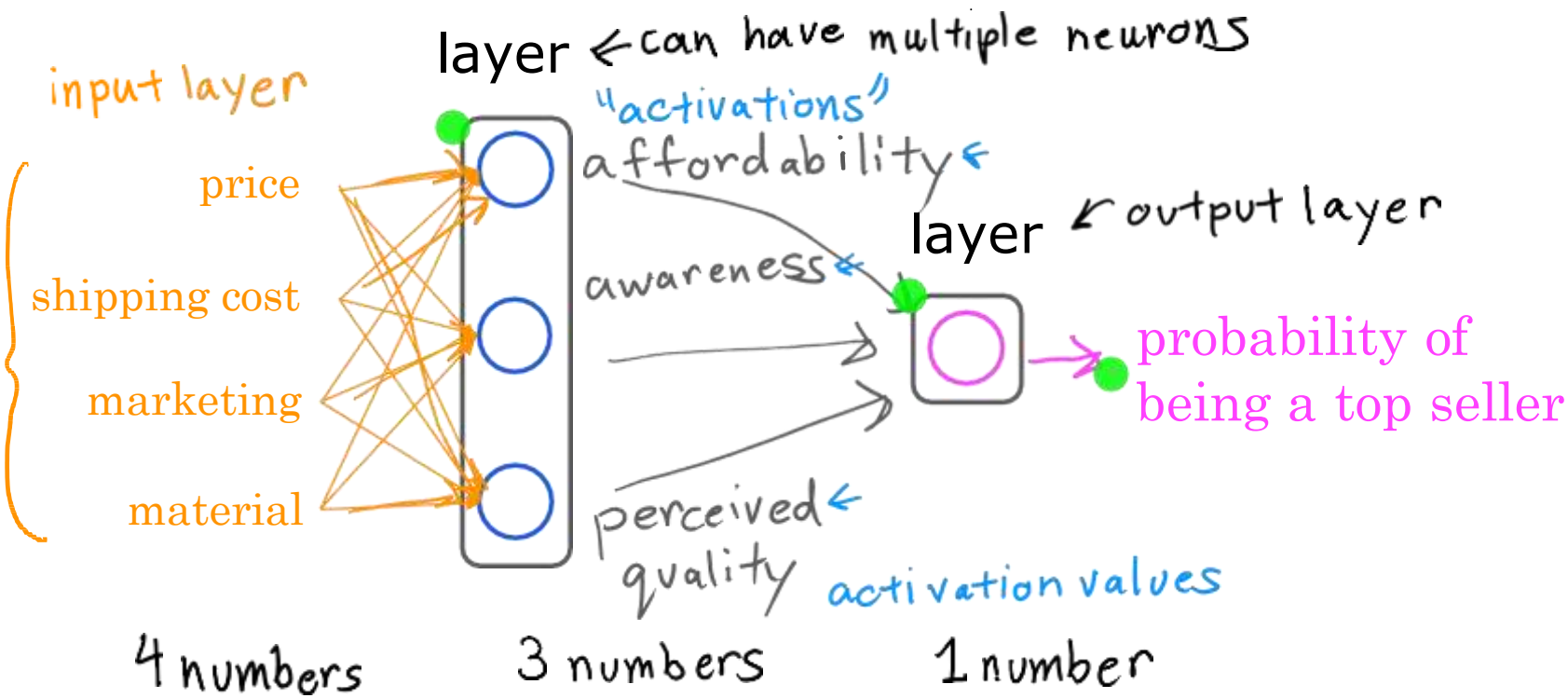
$a = f(x) = \frac{1}{1 + e^{-(wx+b)}}$

output

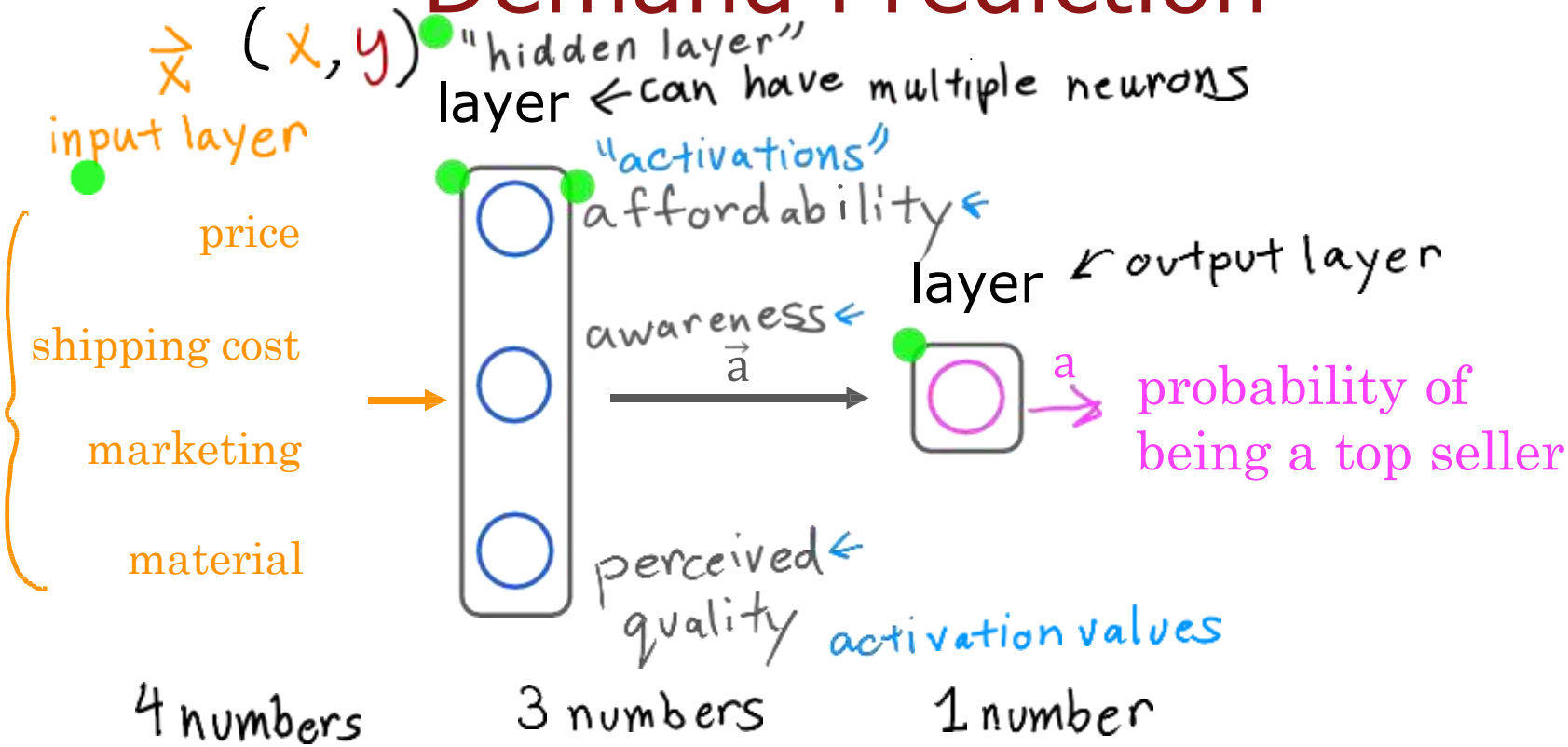
activation



Demand Prediction



Demand Prediction



Demand Prediction

feature engineering

x_1, x_2

$\vec{x}$ (x, y)

input layer

price

shipping cost

marketing

material

4 numbers

"hidden layer"

layer ← can have multiple neurons

"activations"

affordability ←

awareness ←

$\vec{a}$

perceived ← quality

activation values

3 numbers

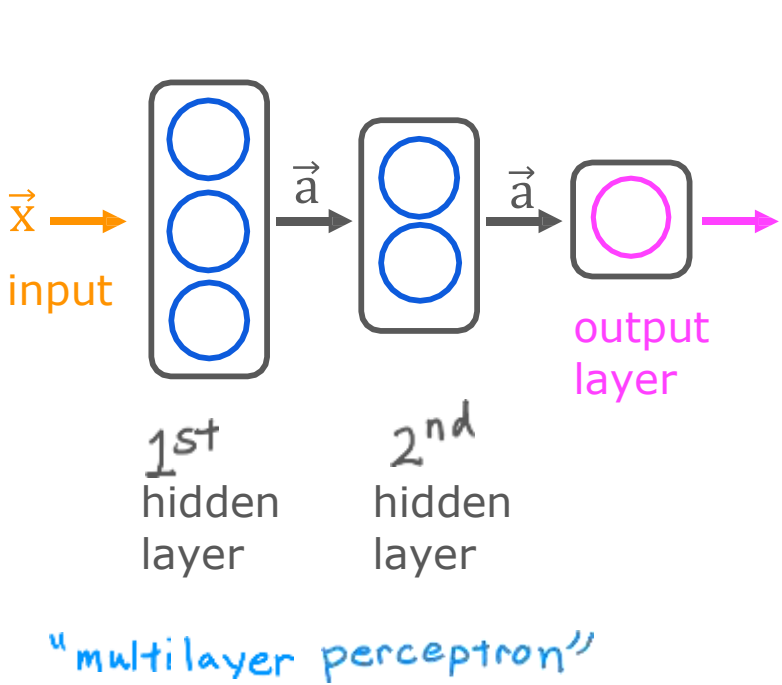
layer ← output layer

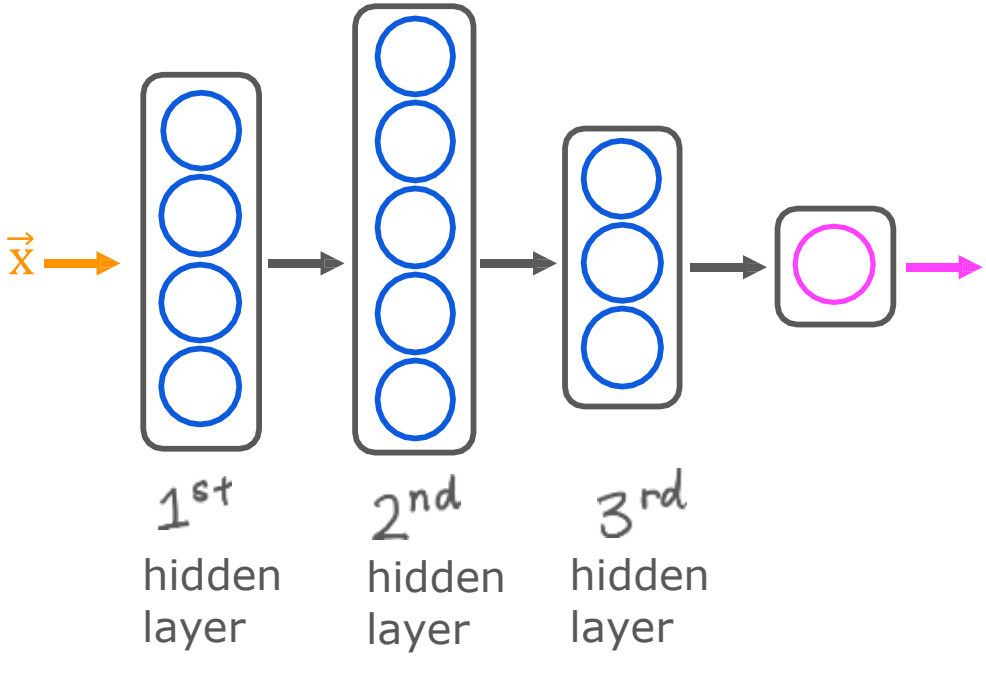
a

probability of being a top seller

1 number

Multiple hidden layers





neural network architecture

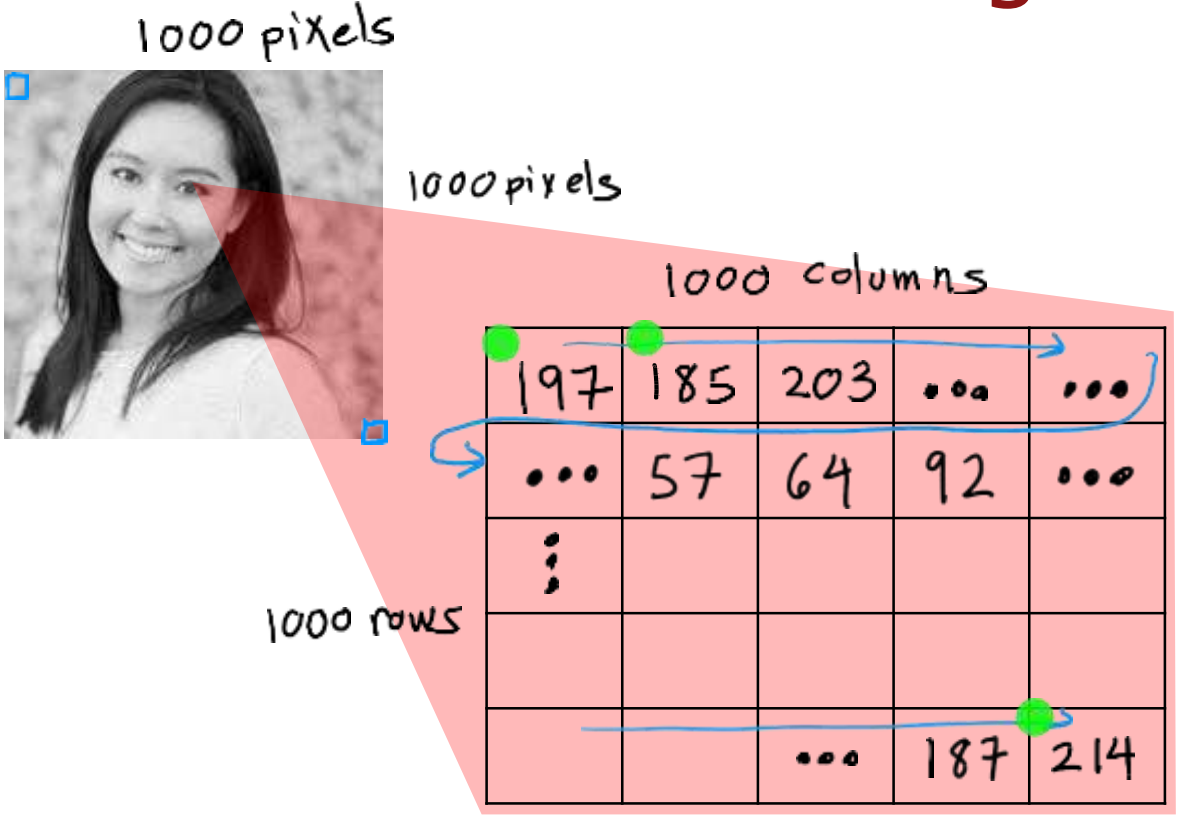
DeepLearning.AI

Stanford
ONLINE

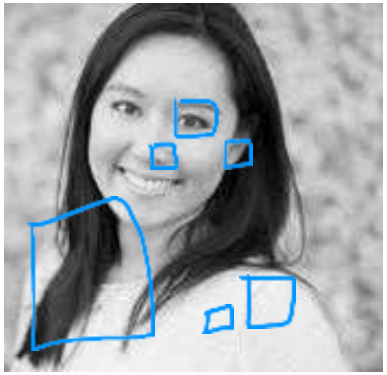
Neural Networks Intuition

Example: Recognizing Images

Face recognition



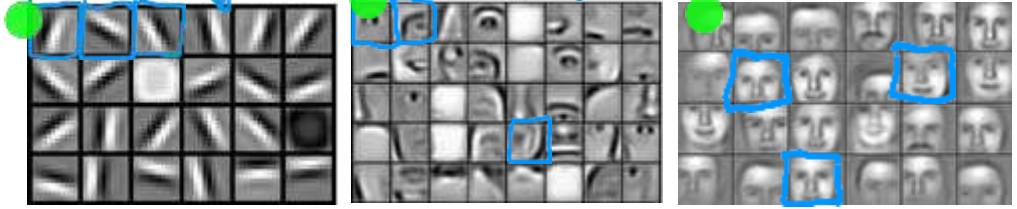
Face recognition



$\vec{x}$
input

output layer

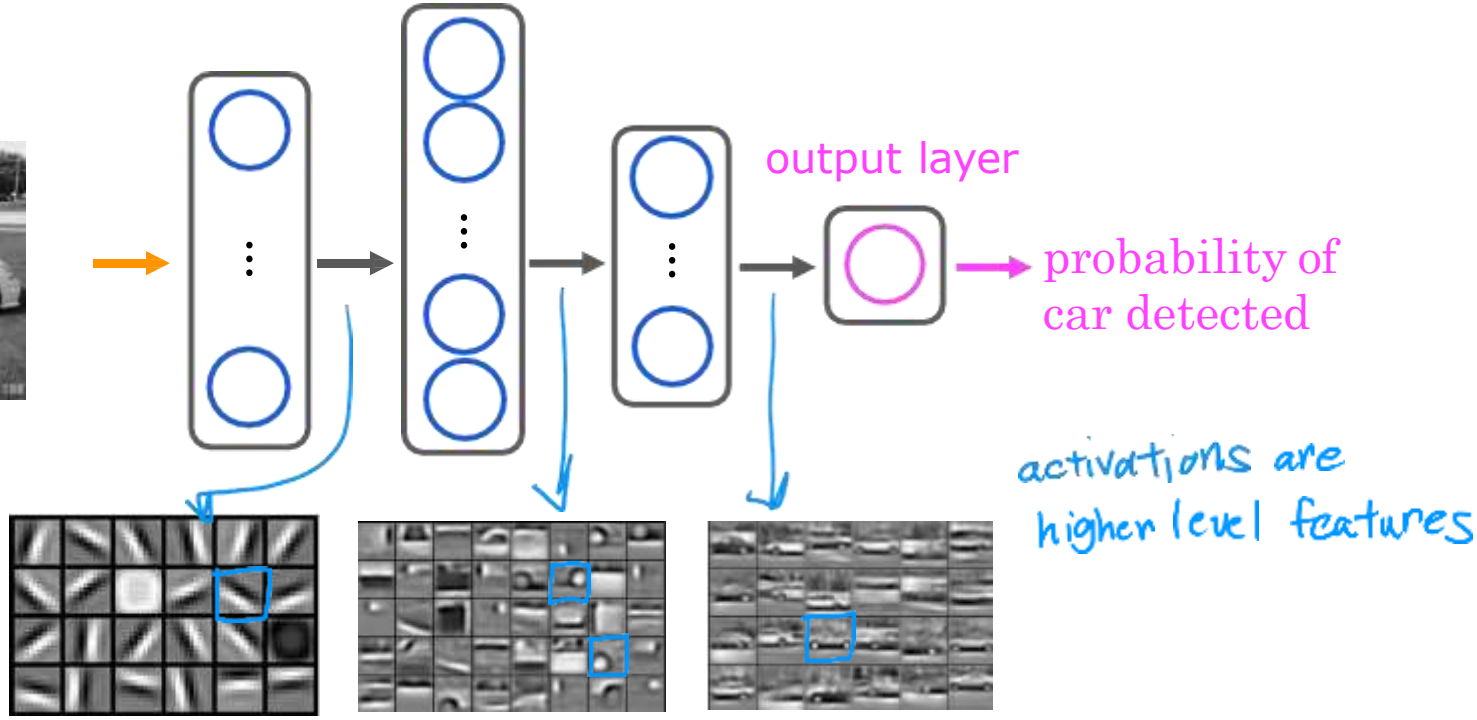
probability of being person 'XYZ'



activations are higher level features

source: Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations by Honglak Lee, Roger Grosse, Ranganath Andrew Y. Ng

Car classification



source: Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations by Honglak Lee, Roger Grosse, Ranganath Andrew Y. Ng

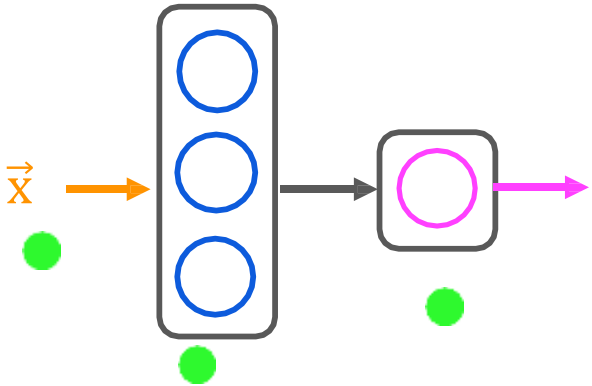
DeepLearning.AI

Stanford
ONLINE

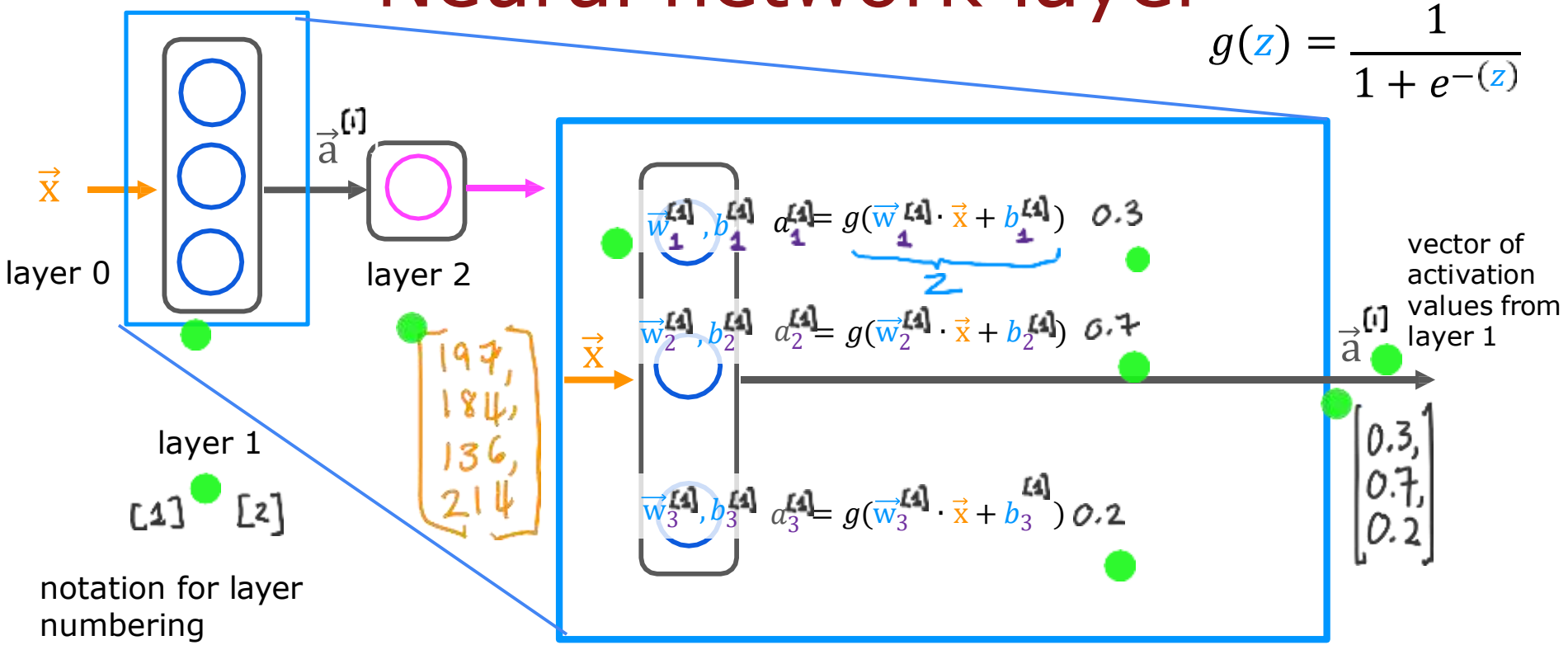
Neural network model

Neural network layer

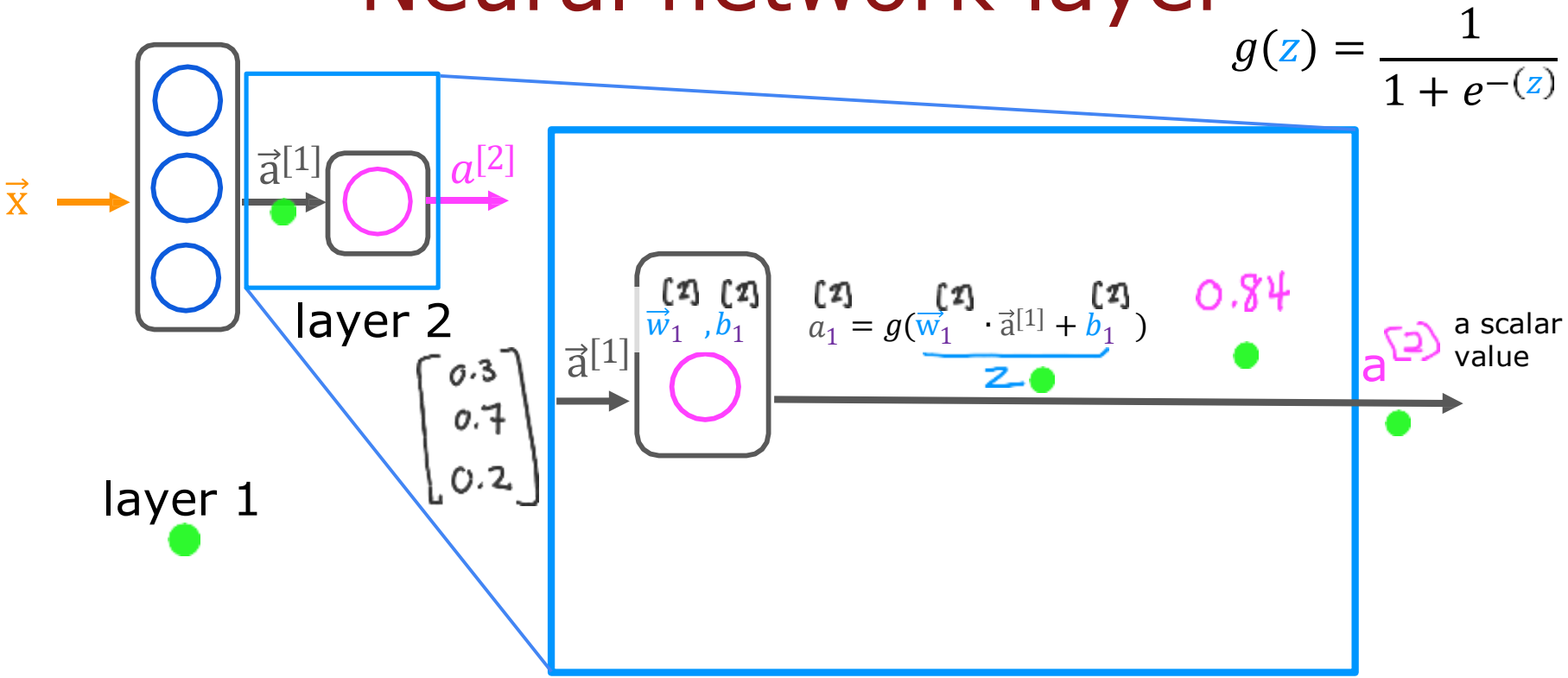
Neural network layer



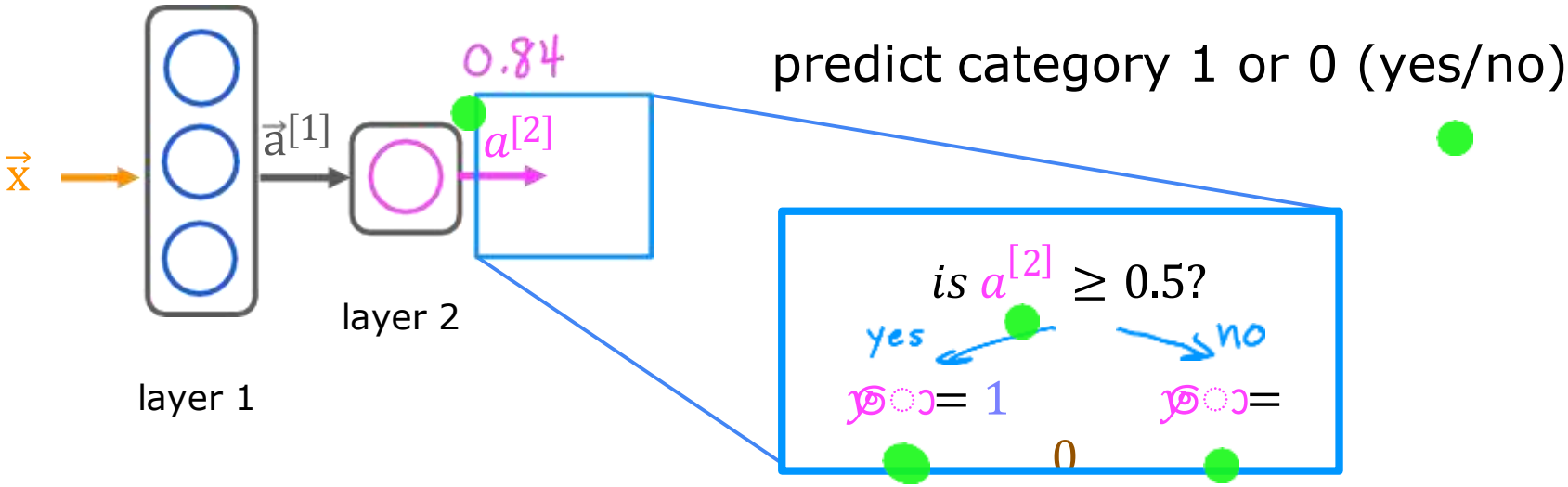
Neural network layer



Neural network layer



Neural network layer



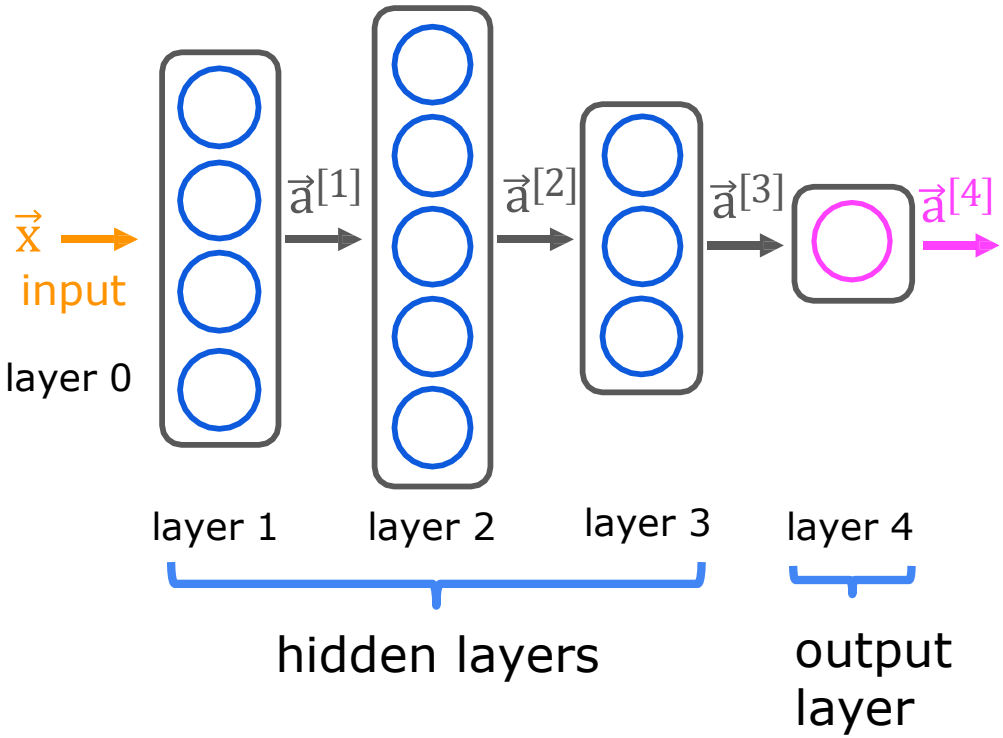
DeepLearning.AI

Stanford
ONLINE

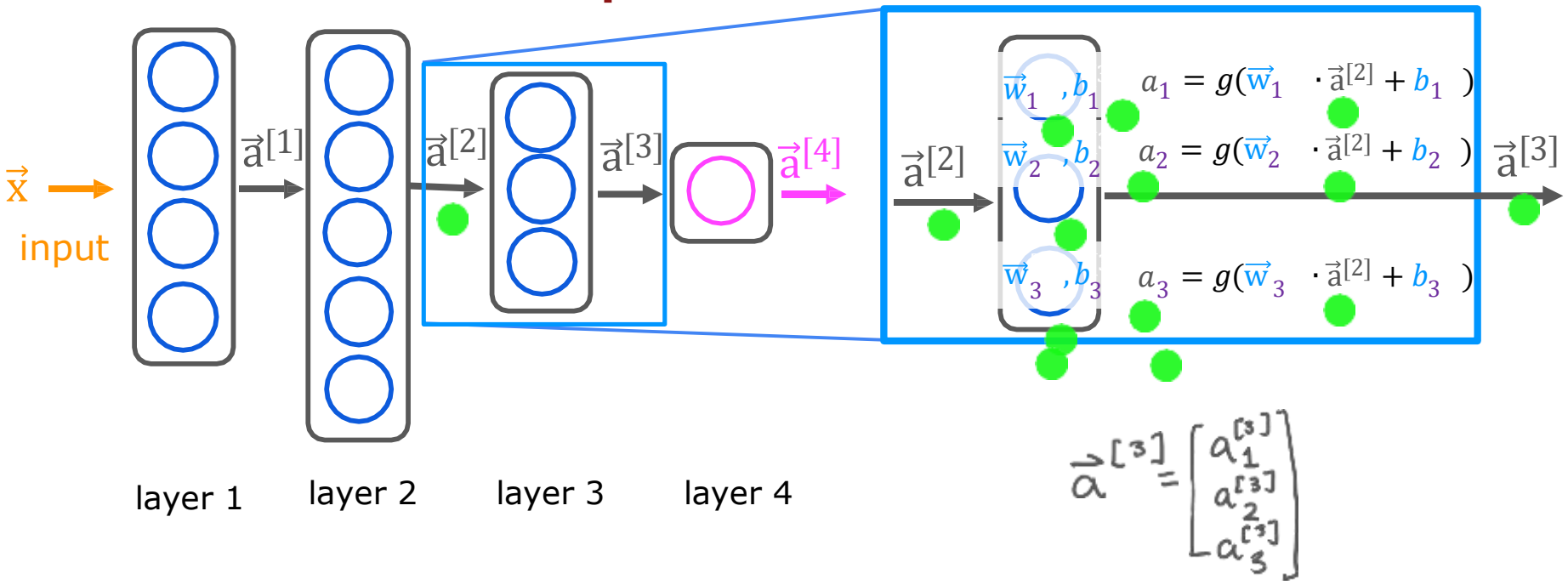
Neural Network Model

More complex neural networks

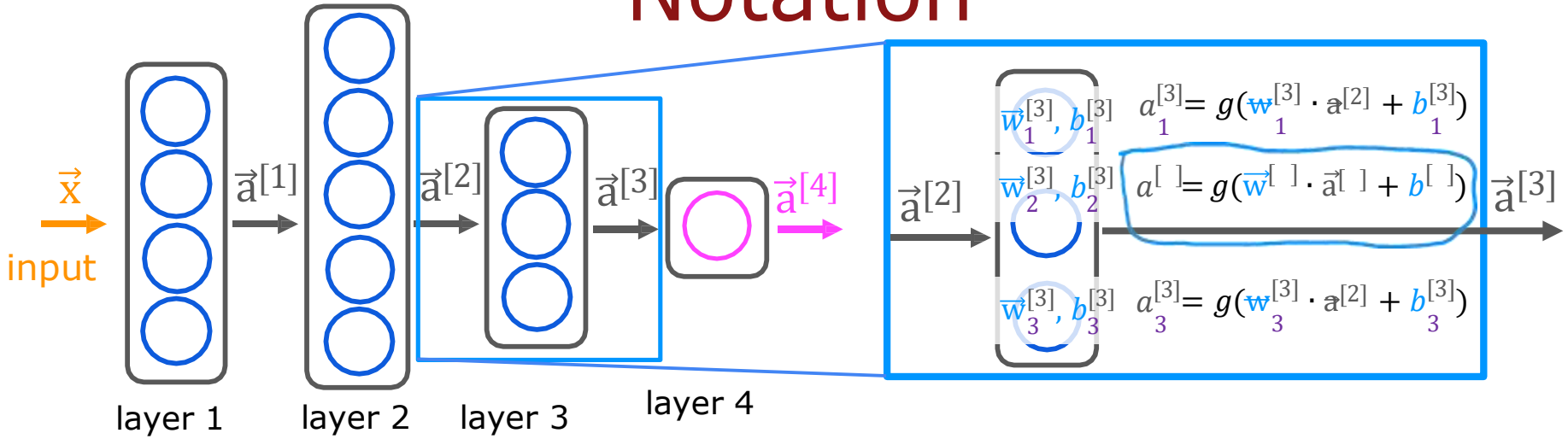
More complex neural network



More complex neural network

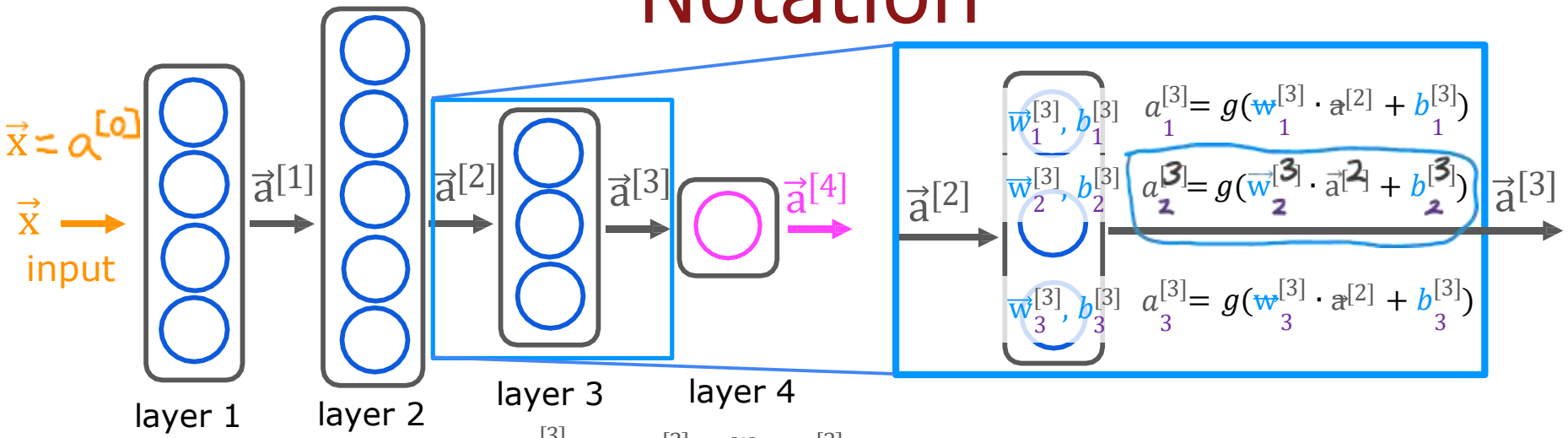


Notation



Question:
Can you fill in the superscripts and subscripts for the second neuron?

Notation



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

Activation value of layer l, unit(neuron) j

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

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

sigmoid "activation function"

Parameters w & b of layer l, unit j

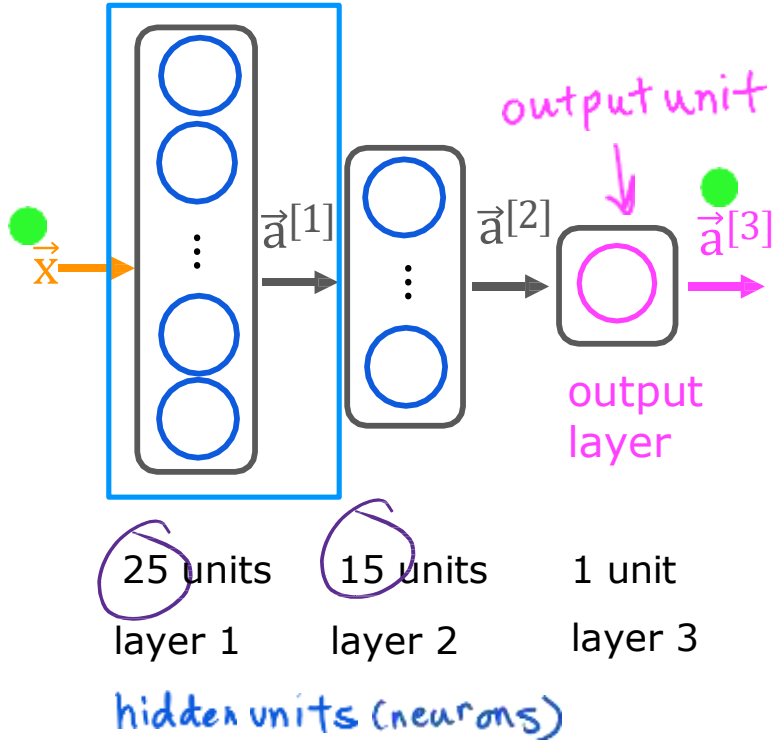
DeepLearning.AI

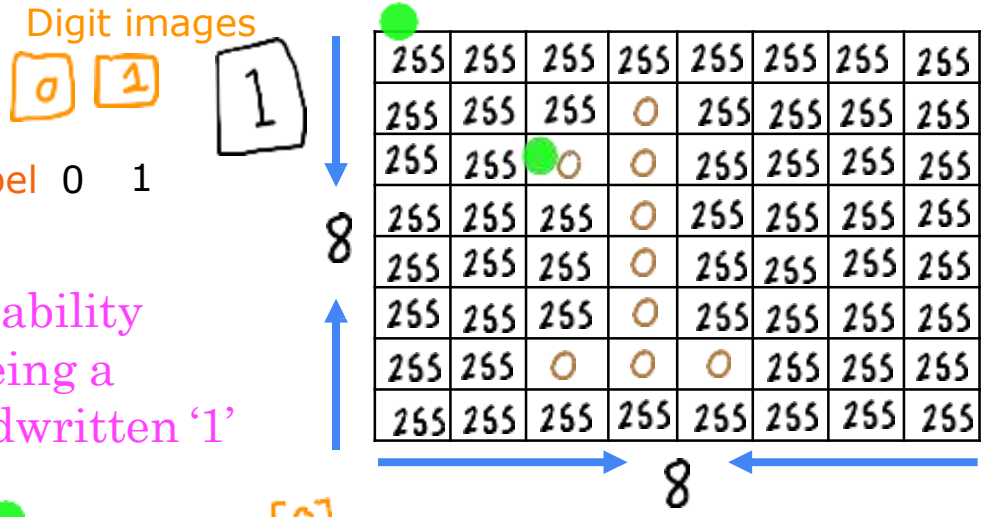
Stanford
ONLINE

Neural Network Model

Inference: making predictions (forward propagation)

Handwritten digit recognition

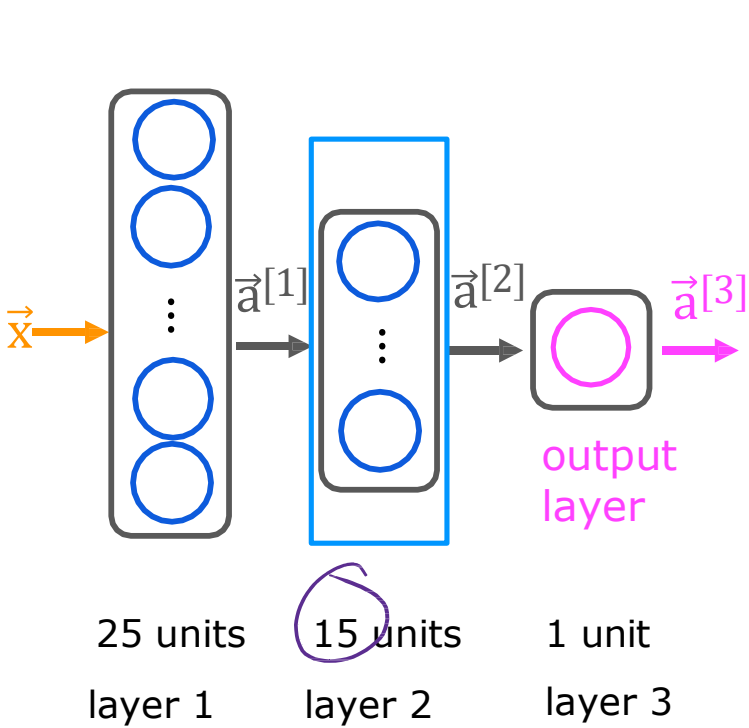


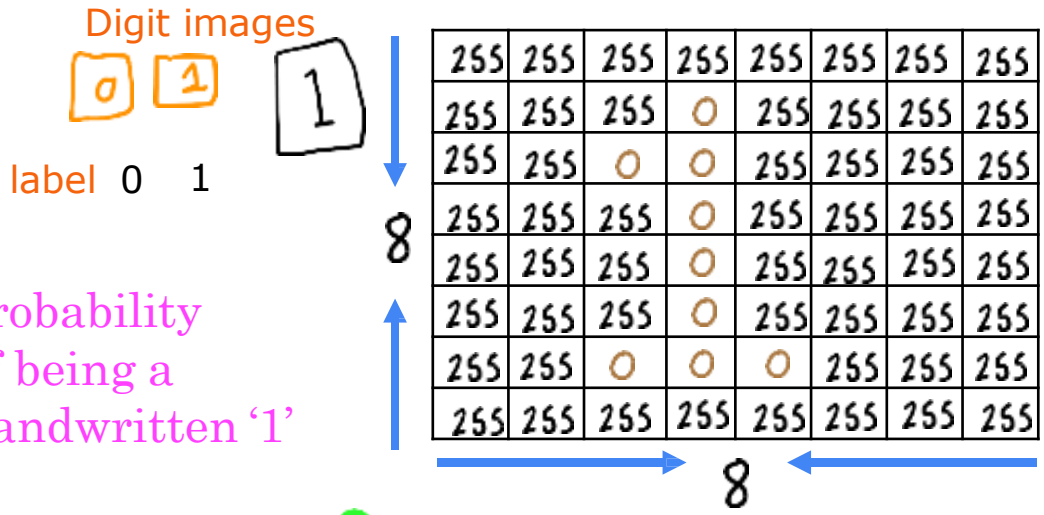


$$a^{[0]}$$

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

Handwritten digit recognition

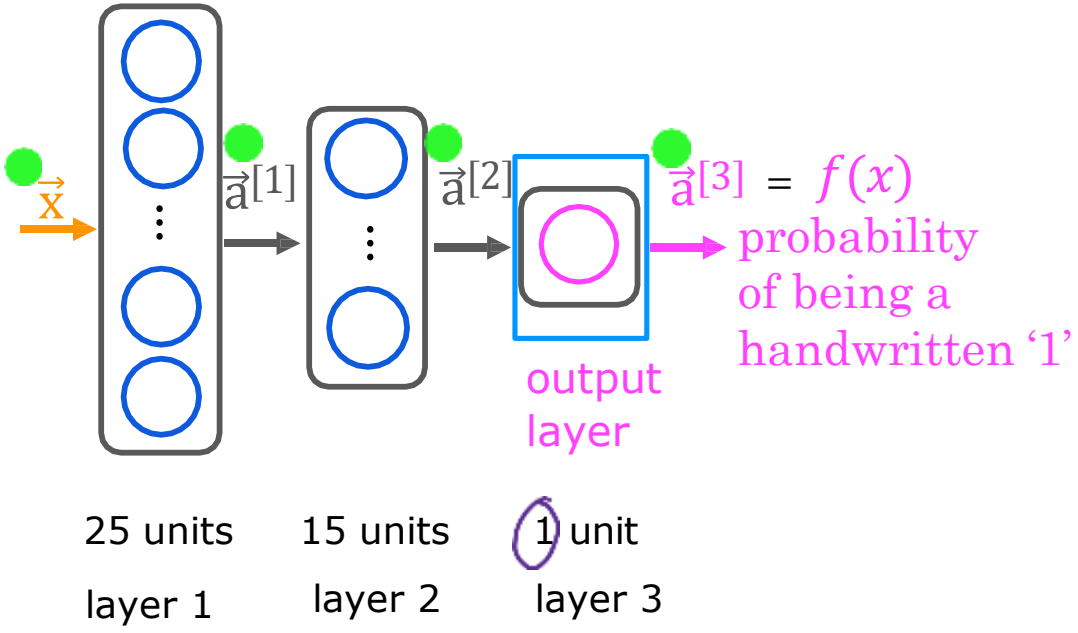




$$\vec{a}^{[2]} = \begin{bmatrix} g(\vec{w}_1^{[2]} \cdot \vec{a}^{[1]} + b_1^{[2]}) \\ \vdots \\ g(\vec{w}_{15}^{[2]} \cdot \vec{a}^{[1]} + b_{15}^{[2]}) \end{bmatrix}$$

Handwritten digit recognition

forward propagation



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

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

yes → $f(x) = 1$ → image is digit 1

no → $f(x) = 0$ → image isn't digit 1

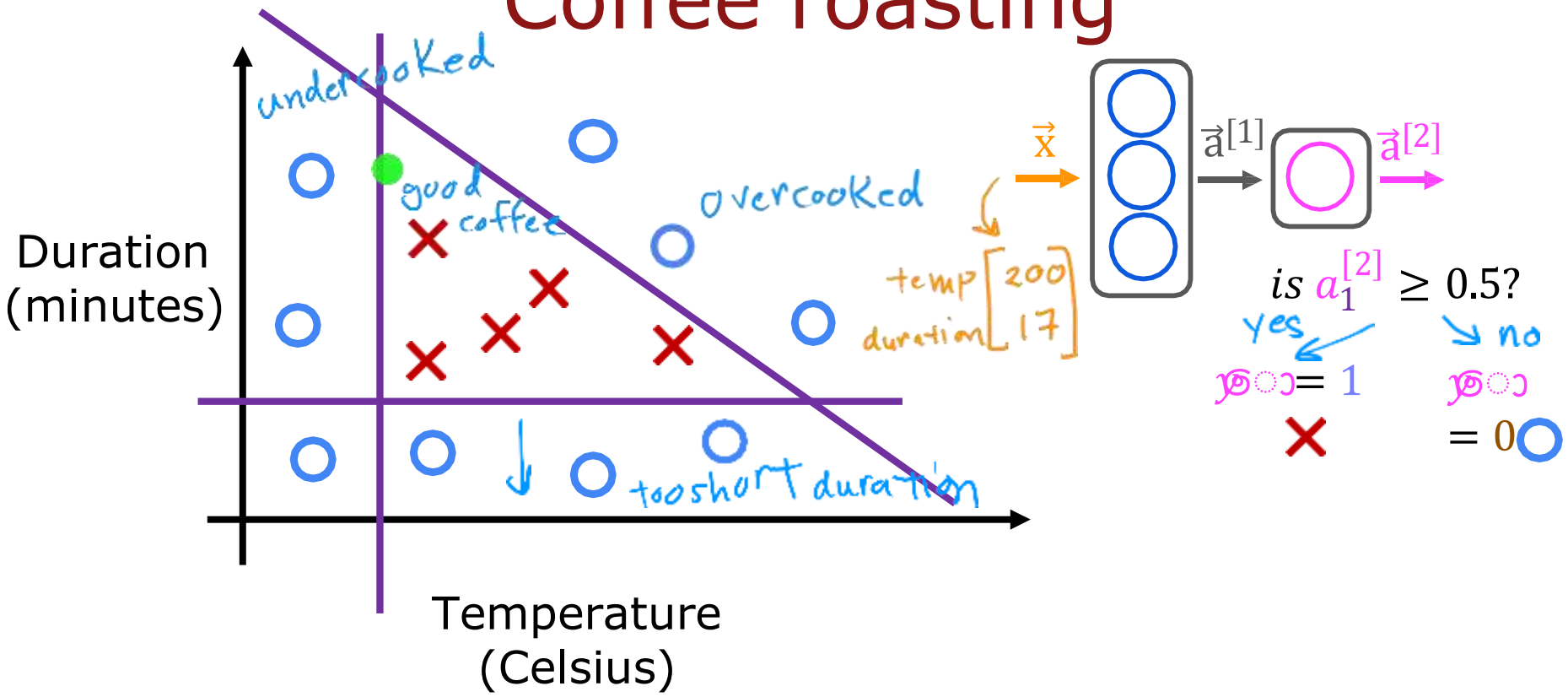
DeepLearning.AI

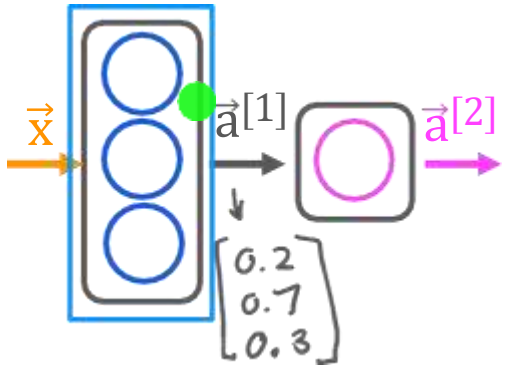
Stanford
ONLINE

TensorFlow implementation

Inference in Code

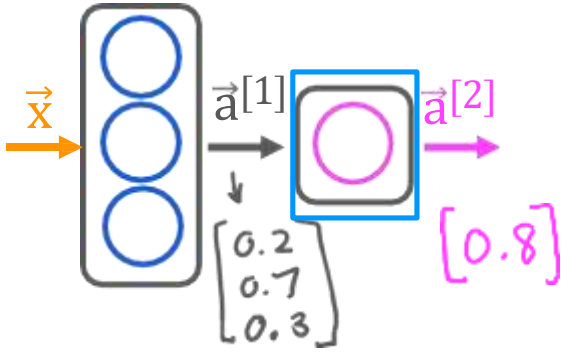
Coffee roasting






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

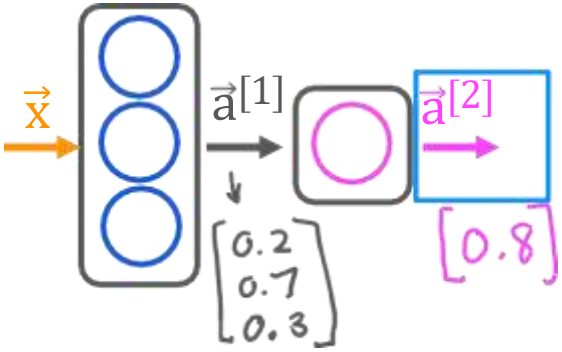
Build the model using TensorFlow

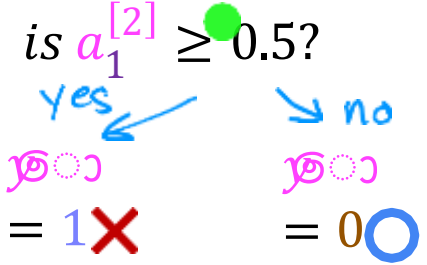



```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation='sigmoid')
a1 = layer_1(x)
```

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

Build the model using TensorFlow



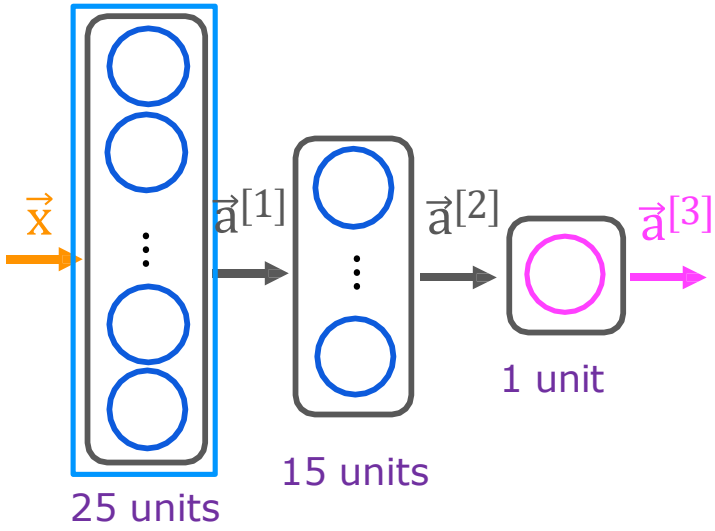



```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation='sigmoid')
a1 = layer_1(x)
```

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

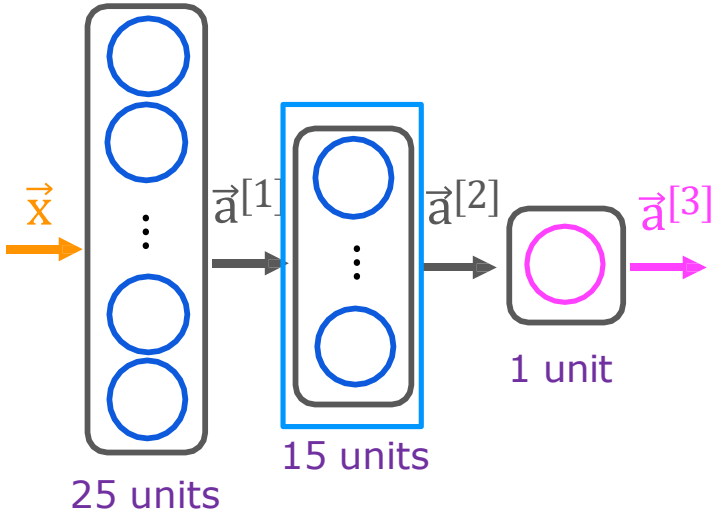
```
if a2 >= 0.5:
  yhat = 1
else:
  yhat = 0
```

Model for digit classification




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

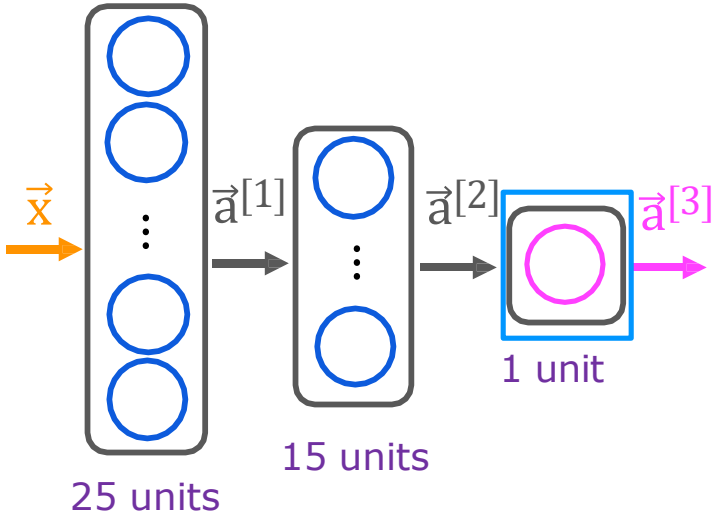
Model for digit classification




```
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)
```

Model for digit classification

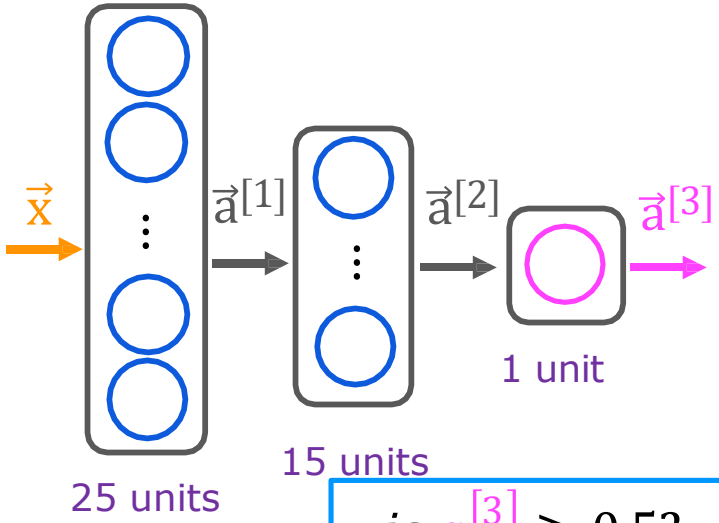



```
x = np.array([[0.0,...245,...240...0]])
layer_1 = Dense(units=25, activation='sigmoid')
a1 = layer_1(x)

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

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

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)
```

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

$= 1$

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

DeepLearning.AI

Stanford
ONLINE

TensorFlow implementation

Data in TensorFlow

Feature vectors

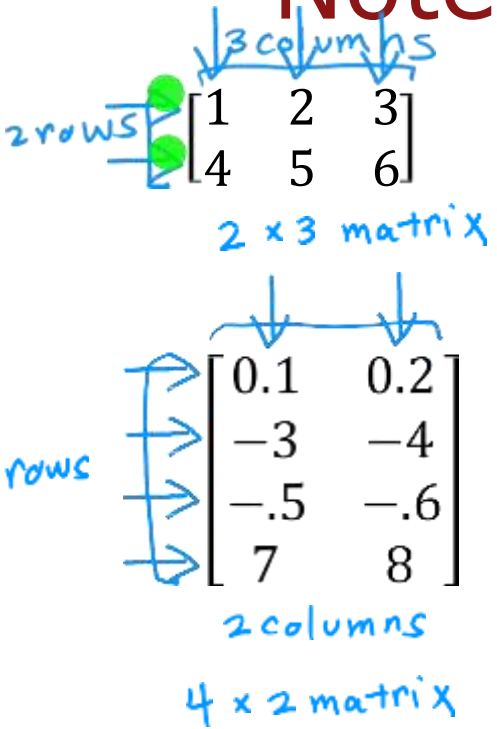
temperature (Celsius)	duration (minutes)	Good coffee? (1/0)
200.0	17.0	1
425.0	18.5	0
…	…	…

```
x = np.array([[200.0, 17.0]])
```

```
[[200.0, 17.0]]
```

Why?

Note about numpy arrays




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

```
[[1, 2, 3],
 [4, 5, 6]]
```

```
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,]]
```

2D array

2 x 3

4 x 2

1 x 2

2 x 1

Note about numpy arrays

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

$[200 \quad 17]$

1 x 2

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

$\begin{bmatrix} 200 \\ 17 \end{bmatrix}$

2 x 1

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

1D
"Vector"

Feature vectors

temperature (Celsius)	duration (minutes)	Good coffee? (1/0)
200.0	17.0	1
425.0	18.5	0
…	…	…

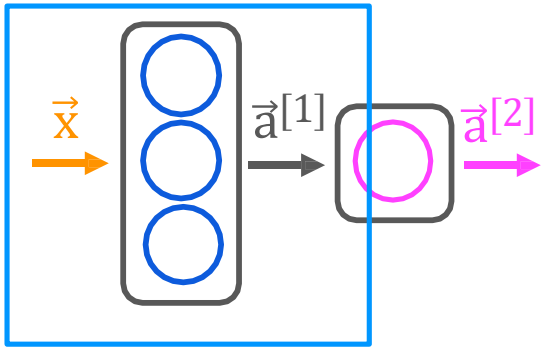
```
x = np.array([[200.0, 17.0]])
```

```
[[200.0, 17.0]]
```

1 x 2

$$\begin{bmatrix} 200.0 & 17.0 \end{bmatrix}$$

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]

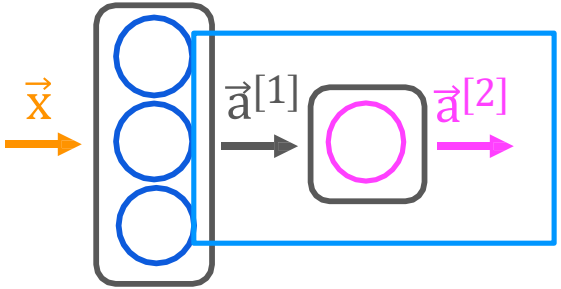
1 x 3 matrix

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

```
a1.numpy()
```

```
array([[1.4661001, 1.125196 , 3.2159438]], 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)
```

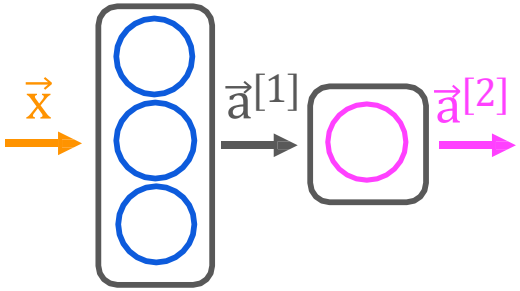
DeepLearning.AI

Stanford
ONLINE

TensorFlow implementation

Building a neural network

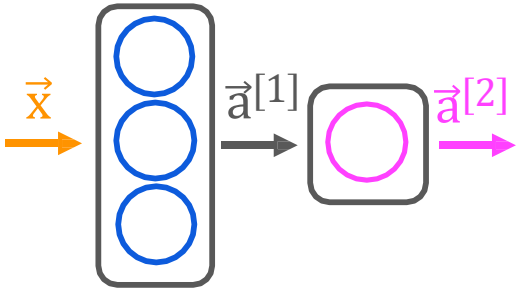
What you saw earlier




```
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)
```

Building a neural network architecture



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

layer1

```
layer_1 = Dense(units=3, activation="sigmoid")
layer_2 = Dense(units=1, activation="sigmoid")
```

layer2

```
model = Sequential([layer_1, layer_2])
```

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

4 x 2

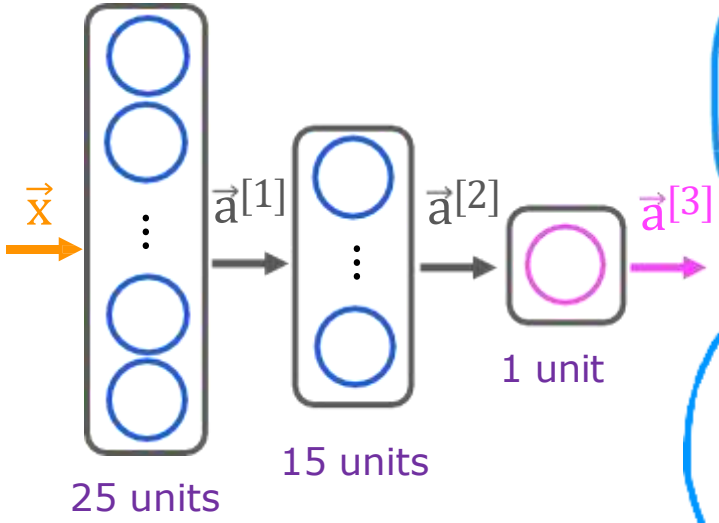
targets

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

more about this next week!

```
model.predict(x_new)
```

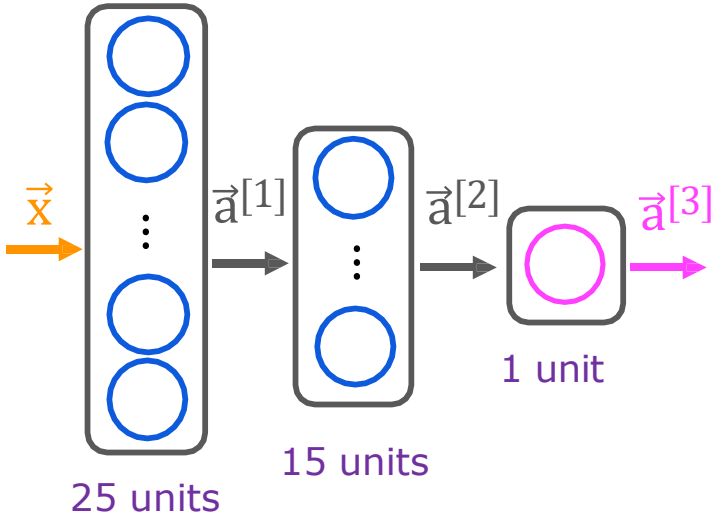
Digit classification model




```
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!

Digit classification model




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

model.compile(...)

x = np.array([[0..., 245, ..., 17],
              [0..., 200, ..., 184]])
y = np.array([1,0])

model.fit(x,y)

model.predict(x_new)
```

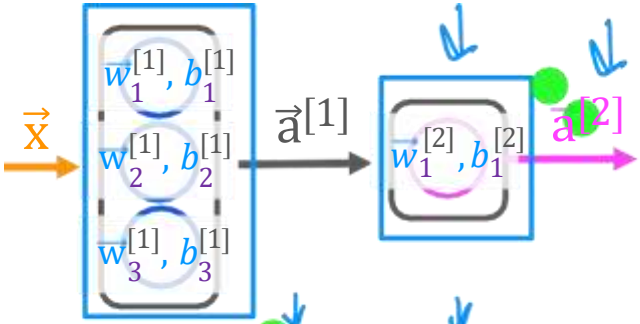
DeepLearning.AI

Stanford
ONLINE

Neural network implementation in Python

Forward prop in a single layer

forward prop (coffee roasting model)



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

```
w2_1 = np.array([-7, 8])
b2_1 = np.array([3])
z2_1 = np.dot(w2_1,a1)+b2_1
a2_1 = sigmoid(z1_1)
```

$w_1^{[2]}$ w2_1

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

1D arrays

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

```
w1_1 = np.array([1, 2])
b1_1 = np.array([-1])
z1_1 = np.dot(w1_1,x)+ b
a1_1 = sigmoid(z1_1)
```

```
w1_2 = np.array([-3, 4])
b1_2 = np.array([1])
z1_2 = np.dot(w1_2,x)+b
a1_2 = sigmoid(z1_2)
```

```
w1_3 = np.array([5, -6])
b1_3 = np.array([2])
z1_3 = np.dot(w1_3,x)+b
a1_3 = sigmoid(z1_3)
```

```
a1   = np.array([a1_1, a1_2, a1_3])
```

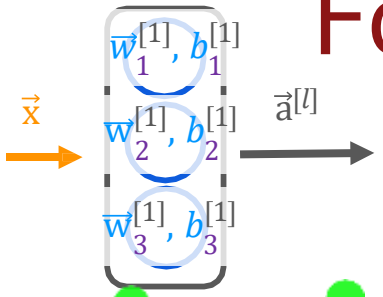
DeepLearning.AI

Stanford
ONLINE

Neural network implemenation in Python

General implementation of forward propagation

Forward prop in NumPy



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

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

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

```
b = np.array([-1, 1, 2])
```

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

```
a_in = 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
```
3 [0,0,0] 0,1,2

```
def sequential(x):
    a1 = dense(x,W1,b1,g)
    a2 = dense(a1,W2,b2,g)
    a3 = dense(a2,W3,b3,g)
    a4 = dense(a3,W4,b4,g)
    f_x = a4
    return f_x
```
$a^{[1]}$ $a^{[2]}$

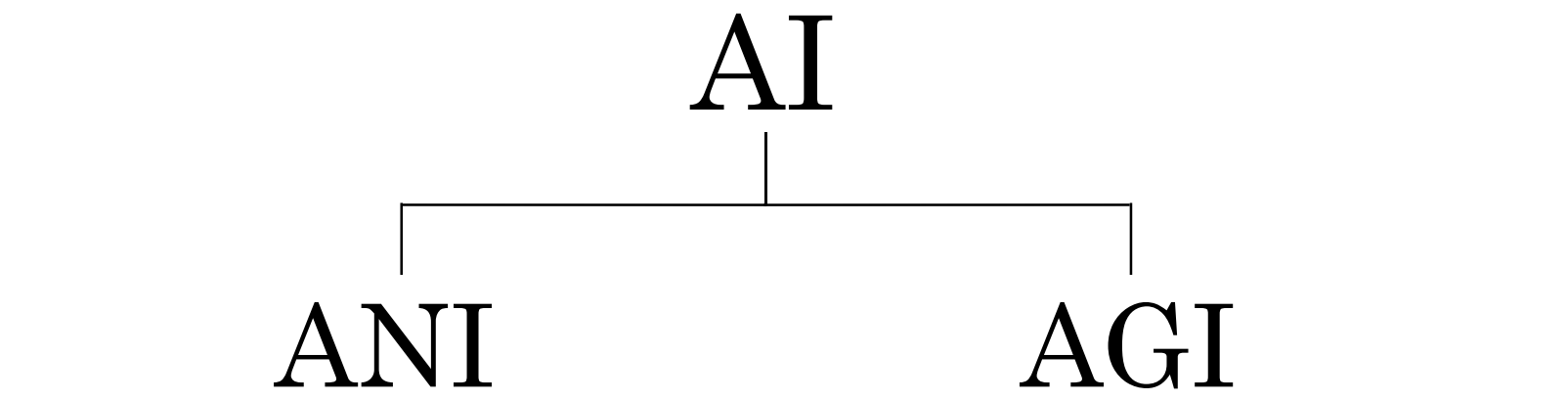
capital W refers to a matrix

DeepLearning.AI

Stanford
ONLINE

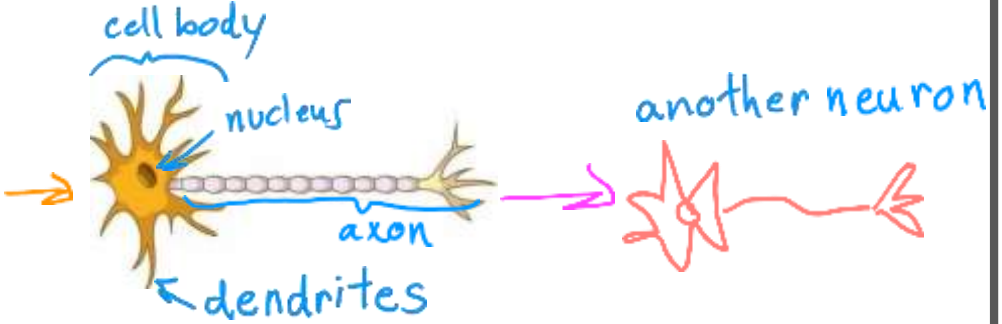
Speculations on artificial general intelligence (AGI)

Is there a path to AGI?


AI
ANI
(artificial narrow intelligence)
E.g., smart speaker,
self-driving car, web search,
AI in farming and factories
AGI
(artificial general intelligence)
Do anything a human can do

Biological neuron

inputs outputs



Simplified mathematical model of a neuron

inputs outputs

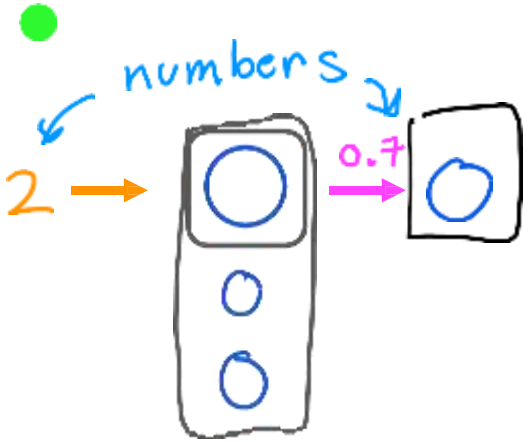
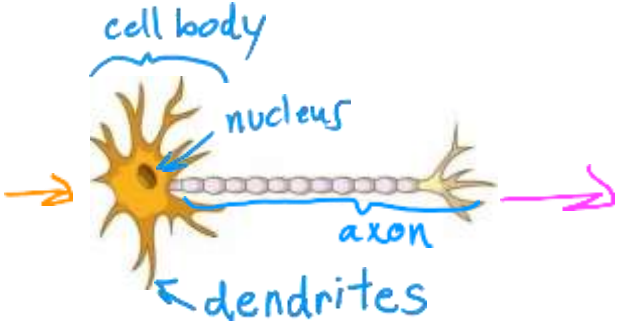


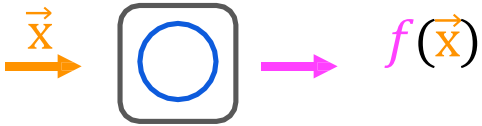
image source: https://biologydictionary.net/sensory-neuron/

Neural network and the brain

Can we mimic the human brain?

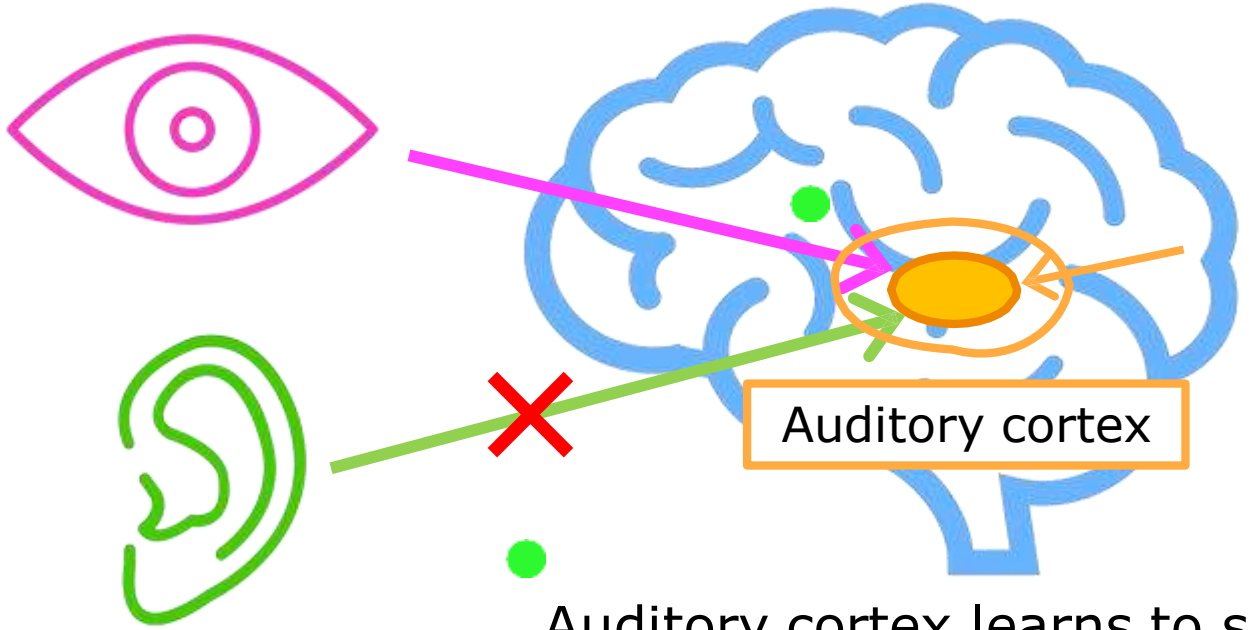


vs



We have (almost) no idea how the brain works

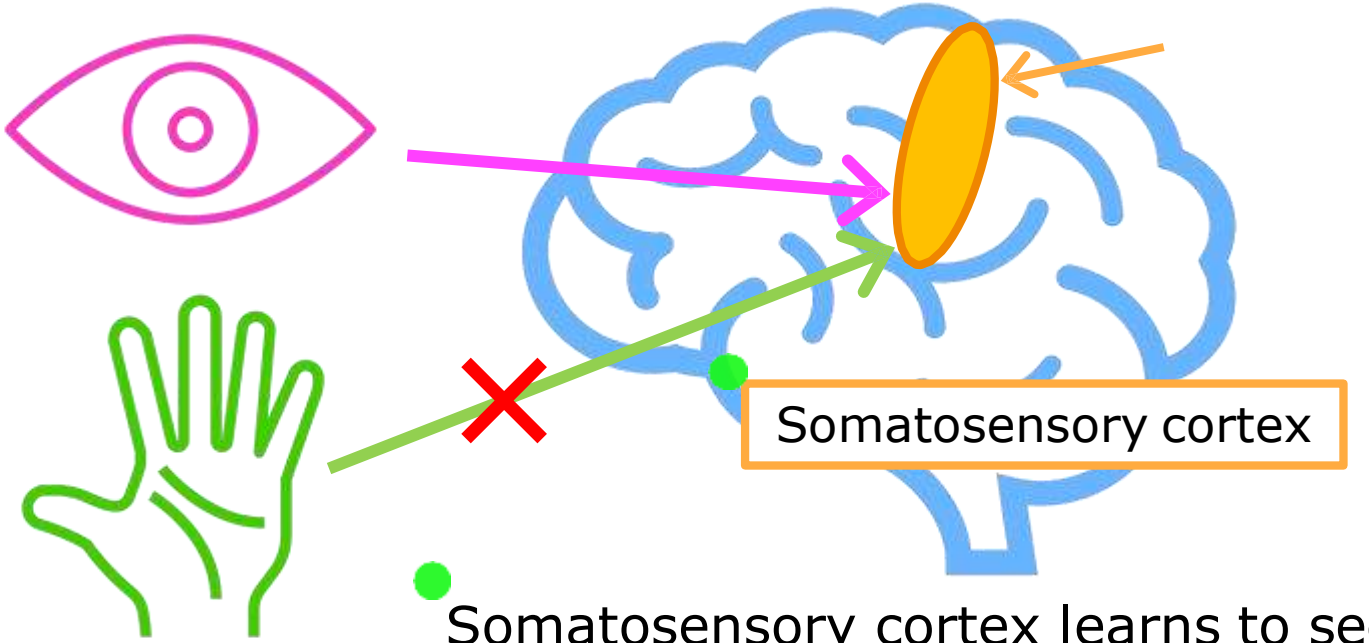
The ”one learning algorithm” hypothesis



Auditory cortex learns to see

[Roe et al., 1992]

The ”one learning algorithm” hypothesis

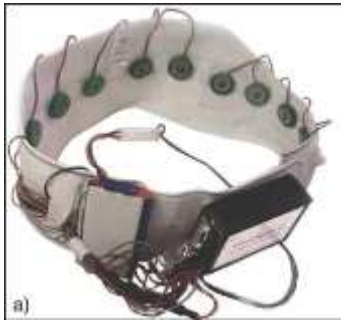


[Metin & Frost, 1989]

Sensor representations in the brain

Seeing with your tongue

Human echolocation (sonar)



Haptic belt: Direction sense

Implanting a 3rd eye

[BrainPort; Welsh & Blasch, 1997; Nagel et al., 2005; Constantine-Paton & Law, 2009]

DeepLearning.AI

Stanford
ONLINE

Vectorization (optional)

How neural networks are implemented efficiently

For loops vs. vectorization

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

W = np.array([[1, -3, 5],
              [-2, 4, -6]])

b = np.array([-1, 1, 2])

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

[1,0,1]

Vectorized

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

W = np.array([[1, -3, 5],
              [-2, 4, -6]])

B = np.array([[-1, 1, 2]])

def dense(A_in,W,B):
    Z = np.matmul(A_in,W) + B
    A_out = g(Z)
  return A_out
```

[[1,0,1]]

2D array

same

1 x 3 2D array

all 2D arrays

matrix multiplication

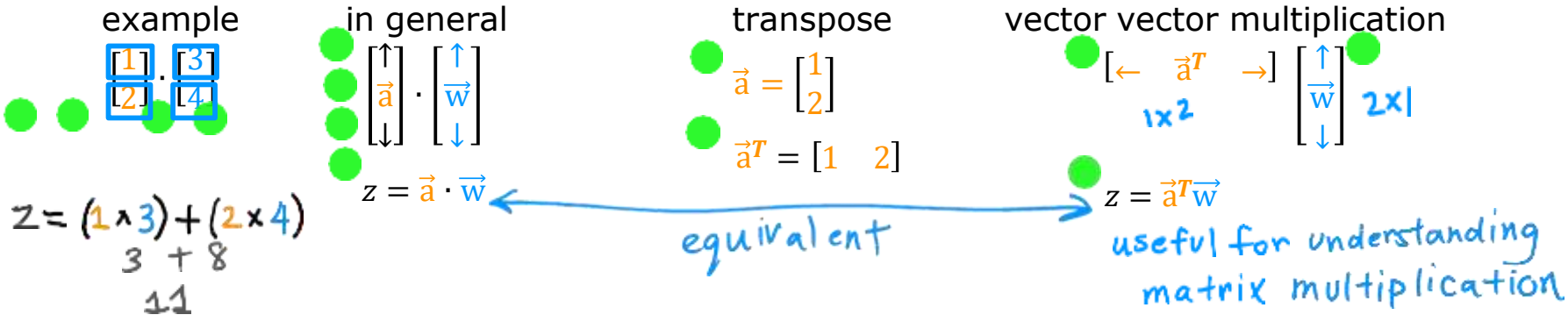
DeepLearning.AI

Stanford
ONLINE

Vectorization
(optional)

Matrix multiplication

Dot products



Vector matrix multiplication

$\vec{a} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

$\vec{a}^T = [1 \quad 2]$ $\quad w = \begin{bmatrix} 3 & 5 \\ 4 & 6 \end{bmatrix}$ $\quad Z = \vec{a}^T w$ 1 by 2 $\quad [\,\rangle \quad \vec{a}^T \rightarrow \begin{bmatrix} \uparrow & \uparrow \\ \vec{w}_1 & \vec{w}_2 \\ \downarrow & \downarrow \end{bmatrix}$

$Z = [\vec{a}^T \vec{w}_1 \quad a^T \vec{w}_2]$

$(1 \times 3) + (2 \times 4)$ $\qquad$ $(1 \times 5) + (2 \times 6)$

$3 + 8$ $\qquad$ $5 + 12$

11 $\qquad$ 17

$Z = [11 \quad 17]$

matrix matrix multiplication

$$A = \begin{bmatrix} 1 & -1 \\ 2 & -2 \end{bmatrix} \quad A^T = \begin{bmatrix} 1 & 2 \\ -1 & -2 \end{bmatrix} \quad W = \begin{bmatrix} 3 & 5 \\ 4 & 6 \end{bmatrix} \quad Z = A^T W = \begin{bmatrix} \rightarrow a_1^T \rightarrow \\ \rightarrow a_2^T \rightarrow \end{bmatrix} \begin{bmatrix} \uparrow & \uparrow \\ \vec{w}_1 & \vec{w}_2 \\ \downarrow & \downarrow \end{bmatrix}$$

rows

columns

$$= \begin{bmatrix} \rightarrow a_1^T \vec{w}_1 & \rightarrow a_1^T \vec{w}_2 \\ \rightarrow a_2^T \vec{w}_1 & \rightarrow a_2^T \vec{w}_2 \end{bmatrix}$$

row 1 col 1

row 2 col 1

row 1 col 2

row 2 col 2

$(-1 \times 3) + (-2 \times 4)$

$-3 + -8$

-11

$(-1 \times 5) + (-2 \times 6)$

$-5 + -12$

-17

$$= \begin{bmatrix} 11 & 17 \\ -11 & -17 \end{bmatrix}$$

general rules for matrix multiplication ↪ next video!

DeepLearning.AI

Stanford
ONLINE

Vectorization
(optional)

Matrix multiplication rules

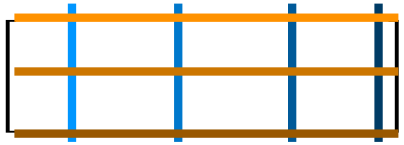
Matrix multiplication rules

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 0.1 \\ 2 & -2 & 0.2 \end{bmatrix} \quad \mathbf{A}^T = \begin{bmatrix} 1 & 2 \\ -1 & -2 \\ 0.1 & 0.2 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 3 & 5 & 7 & 9 \\ 4 & 6 & 8 & 0 \end{bmatrix}$$

$\vec{a}_1^T$ $\vec{a}_2^T$ $\vec{a}_3^T$

$\vec{w}_1$ $\vec{w}_2$ $\vec{w}_3$ $\vec{w}_4$

$$Z = \mathbf{A}^T\mathbf{W} =$$



Matrix multiplication rules

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 0.1 \\ 2 & -2 & 0.2 \end{bmatrix} \quad \mathbf{A}^T = \begin{bmatrix} 1 & 2 \\ -1 & -2 \\ 0.1 & 0.2 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 3 & 5 & 7 & 9 \\ 4 & 6 & 8 & 0 \end{bmatrix} \quad Z = \mathbf{A}^T\mathbf{W} = \begin{bmatrix} & & & \\ & & & \\ & & & \end{bmatrix}$$

3 by 4 matrix

$$\vec{a}_1^T \vec{w}_1 = (1 \times 3) + (2 \times 4) = 11$$

row 3 column 2

$$\vec{a}_3^T \vec{w}_2 = (0.1 \times 5) + (0.2 \times 6) = 1.7$$

$$0.5 \quad + \quad 1.2$$

row 2 column 3?

$$\vec{a}_2^T \vec{w}_3 = (-1 \times 7) + (-2 + 8) = -23$$

$$-7 \quad + \quad -16$$

Matrix multiplication rules

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 0.1 \\ 2 & -2 & 0.2 \end{bmatrix} \quad \mathbf{A}^T = \begin{bmatrix} 1 & 2 \\ -1 & -2 \\ 0.1 & 0.2 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 3 & 5 & 7 & 9 \\ 4 & 6 & 8 & 0 \end{bmatrix} \quad \mathbf{Z} = \mathbf{A}^T\mathbf{W} = \begin{bmatrix} 11 & 17 & 23 & 9 \\ -11 & -17 & -23 & -9 \\ 1.1 & 1.7 & 2.3 & 0.9 \end{bmatrix}$$

3 by 4 matrix

$$\vec{a}_1^T \vec{w}_1 = (1 \times 3) + (2 \times 4) = 11$$

row 3 column 2

$$\vec{a}_3^T \vec{w}_2 = (0.1 \times 5) + (0.2 \times 6) = 1.7$$

$0.5 + 1.2$

row 2 column 3?

$$\vec{a}_2^T \vec{w}_3 = (-1 \times 7) + (-2 + 8) = -23$$

$-7 + -16$

Matrix multiplication rules

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 0.1 \\ 2 & -2 & 0.2 \end{bmatrix} \quad \mathbf{A}^T = \begin{bmatrix} 1 & 2 \\ -1 & -2 \\ 0.1 & 0.2 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 3 & 5 & 7 & 9 \\ 4 & 6 & 8 & 0 \end{bmatrix}$$

$$\boldsymbol{Z} = \mathbf{A}^T\mathbf{W} = \begin{bmatrix} 11 & 17 & 23 & 9 \\ -11 & -17 & -23 & -9 \\ 1.1 & 1.7 & 2.3 & 0.9 \end{bmatrix}$$

3 x 2 2 x 4

can only take dot products of vectors that are same length

[0.1 0.2] length 2

$\begin{bmatrix} 5 \\ 6 \end{bmatrix}$ length 2

3 by 4 matrix

↳ same # rows as A^T

same # columns as W

DeepLearning.AI

Stanford
ONLINE

Vectorization
(optional)

Matrix multiplication code

Matrix multiplication in NumPy

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 0.1 \\ 2 & -2 & 0.2 \end{bmatrix} \quad \mathbf{A}^T = \begin{bmatrix} 1 & 2 \\ -1 & -2 \\ 0.1 & 0.2 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 3 & 5 & 7 & 9 \\ 4 & 6 & 8 & 0 \end{bmatrix} \quad \mathbf{Z} = \mathbf{A}^T\mathbf{W} = \begin{bmatrix} 11 & 17 & 23 & 9 \\ -11 & -17 & -23 & -9 \\ 1.1 & 1.7 & 2.3 & 0.9 \end{bmatrix}$$

```
A=np.array([1,-1,0.1],
           [2,-2,0.2]])
```

```
W=np.array([3,5,7,9],
           [4,6,8,0])
```

```
Z = np.matmul(AT,W)
```

or

```
Z = AT @ W
```

```
AT=np.array([1,2],
            [-1,-2],
            [0.1,0.2])
```

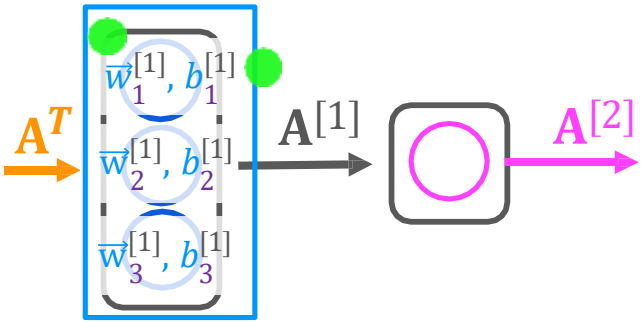
```
AT=A.T
```

transpose

result

```
[[11,17,23,9],
 [-11,-17,-23,-9],
 [1.1,1.7,2.3,0.9]
]
```

Dense layer vectorized



$\mathbf{A}^T = [200 \quad 17]$ (1 x 2)

$\mathbf{W} = \begin{bmatrix} 1 & -3 & 5 \\ -2 & 4 & -6 \end{bmatrix}$ (2 x 3)

$\mathbf{b} = [-1 \quad 1 \quad 2]$ (1 x 3)

$\mathbf{Z} = \mathbf{A}^T\mathbf{W} + \mathbf{B}$

$[165 \quad -531 \quad 900]$

$z_1^{[1]} \quad z_2^{[1]} \quad z_3^{[1]}$

$\mathbf{A} = g(\mathbf{Z})$

$[1 \quad 0 \quad 1]$

A

```
AT = np.array([[200, 17]])
W = np.array([[1, -3, 5],
              [-2, 4, -6]])
b = np.array([[-1, 1, 2]])
```

a_in

```
def dense(AT,W,b,g):
  z = np.matmul(AT,W) + b
  a_out = g(z)
  return a_out
```

a_in

[[1,0,1]]

Copyright Notice

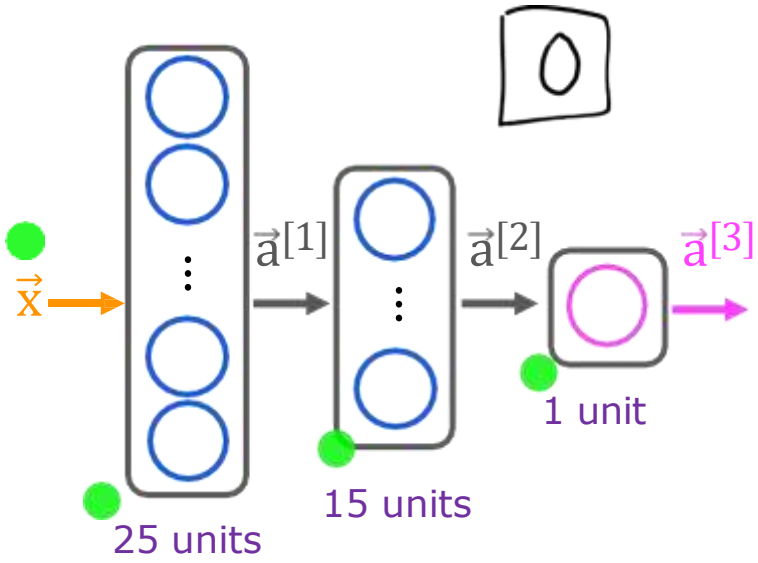
DeepLearning.AI

Stanford
ONLINE

Neural Network Training

TensorFlow implementation

Train a Neural Network in TensorFlow




```
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')
                              )]
from tensorflow.keras.losses import
BinaryCrossentropy
   model.compile(loss=BinaryCrossentropy())

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

1

2

3

epochs: number of steps in gradient descent

Given set of (x,y) examples

How to build and train this in code?

DeepLearning.AI

Stanford ONLINE

Neural Network Training

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

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

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

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

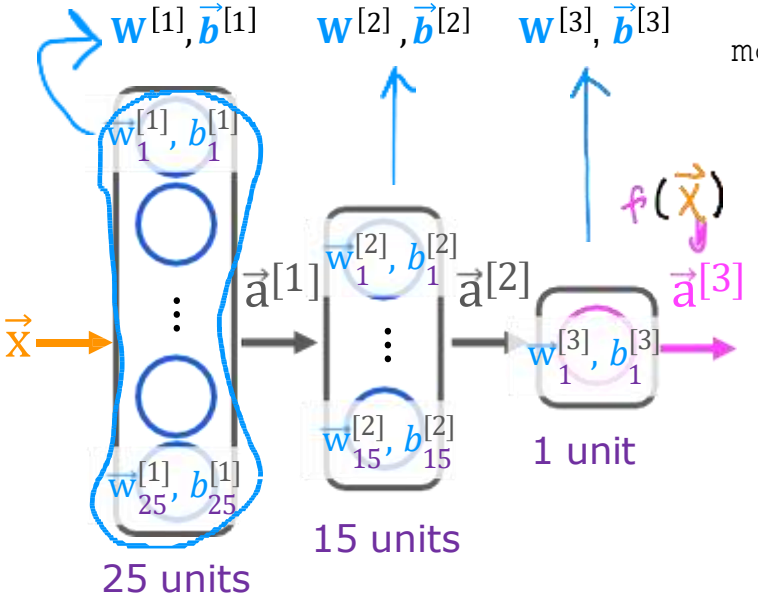
```
model.compile(
loss=BinaryCrossentropy())
```

```
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

Mnist digit classification problem

binary classification

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

Compare prediction vs. target

logistic loss

also known as binary cross entropy

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

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

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

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

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

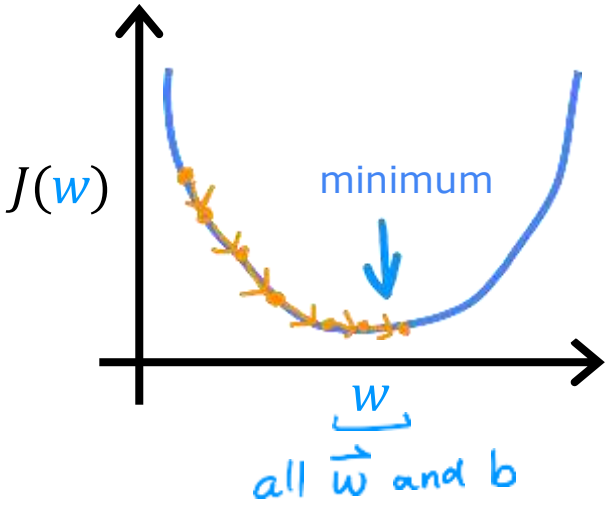
regression (predicting numbers and not categories)

mean squared error

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

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

3. Gradient descent



repeat {

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

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

}

Compute derivatives for gradient descent using "back propagation"

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

Neural network libraries

Use code libraries instead of coding ”from scratch”

Good to understand the implementation
(for tuning and debugging).

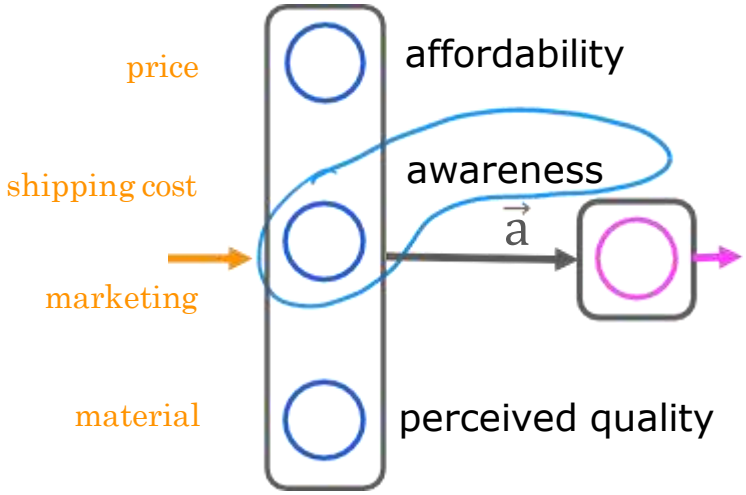
DeepLearning.AI

Stanford ONLINE

Activation Functions

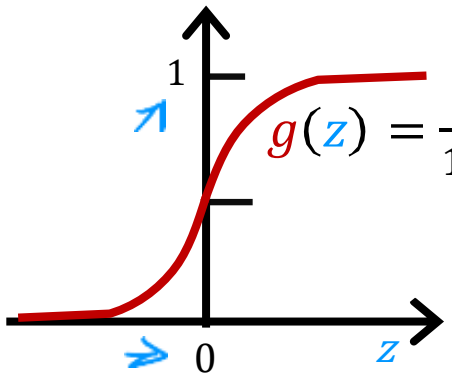
Alternatives to the sigmoid activation

Demand Prediction Example



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

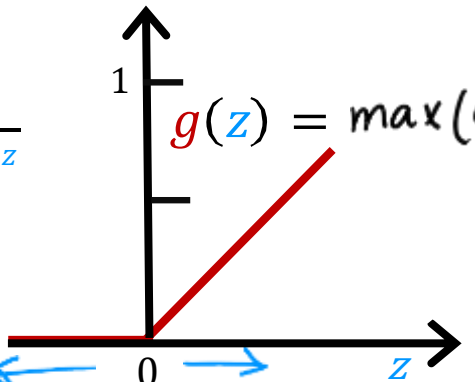
Sigmoid



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

$$0 < g(z) < 1$$

ReLU Rectified Linear Unit



$$g(z) = \max(0, z)$$

if $z < 0$, $g(z)$ is 0

if $z \geq 0$, $g(z)$ is z

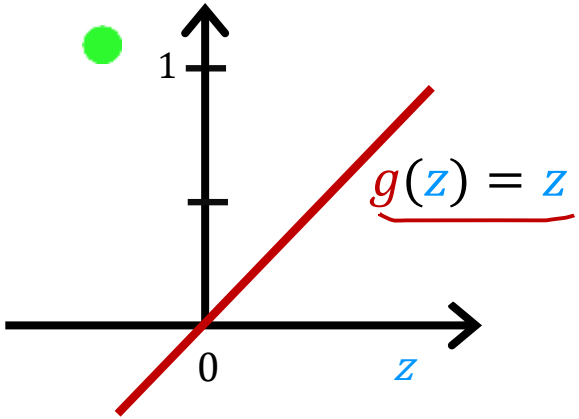
Examples of Activation Functions

"No activation function"

$a^{[1]}_2 = g(\underbrace{w^{[1]}_2 \cdot x + b^{[1]}_2}_{z})$

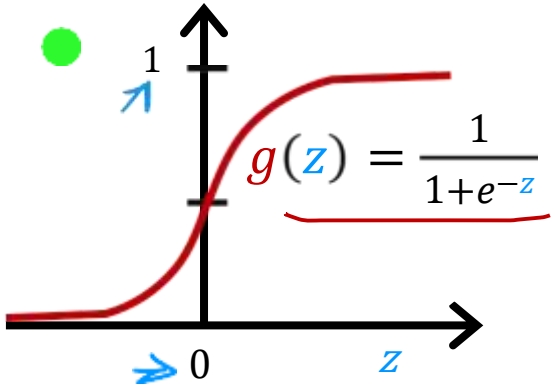
Later: softmax activation

Linear activation function



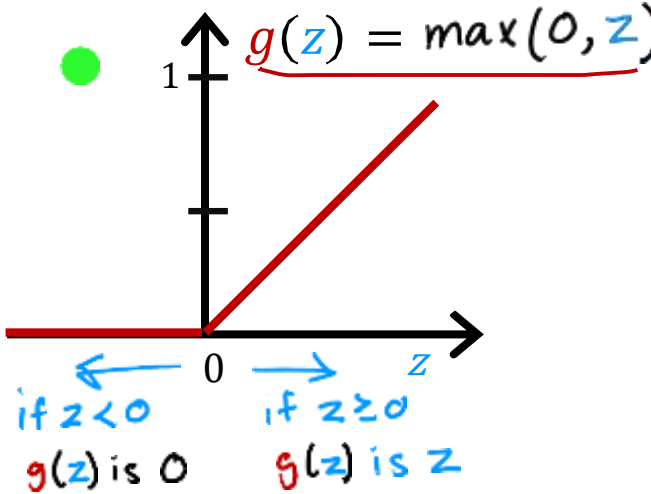
$a = g(z) = \underbrace{\vec{w} \cdot \vec{x} + b}_{z}$

Sigmoid



$0 < g(z) < 1$

ReLU Rectified Linear Unit



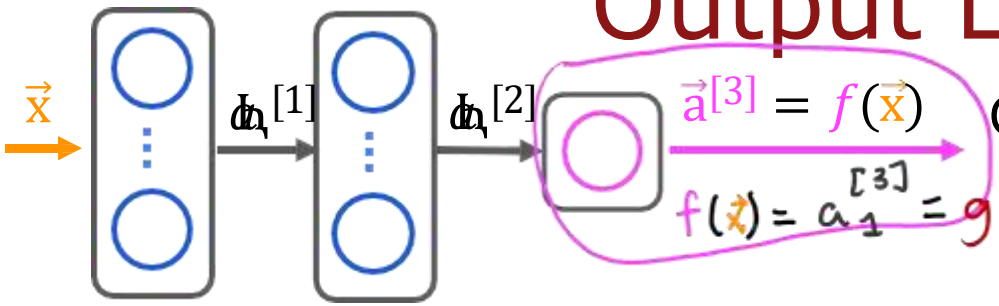
DeepLearning.AI

Stanford ONLINE

Activation Functions

Choosing activation functions

Output Layer

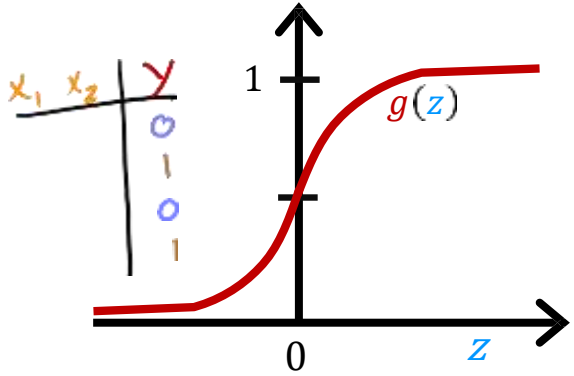


Choosing $g(z)$ for output layer?

Binary classification

Sigmoid

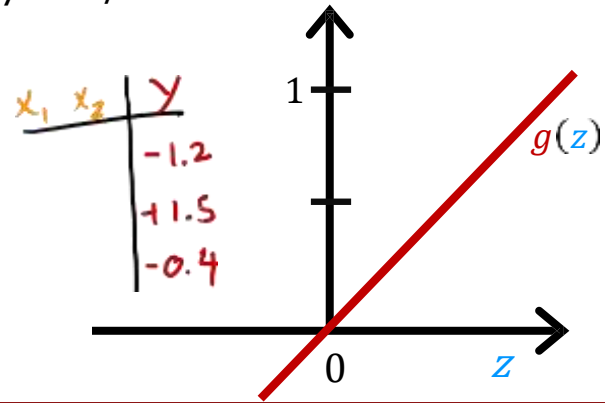
y=0/1



Regression

Linear activation function

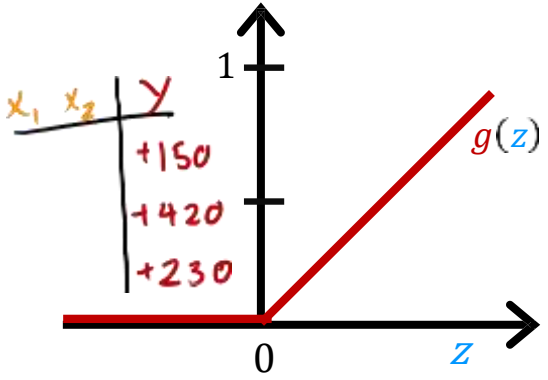
y = +/-



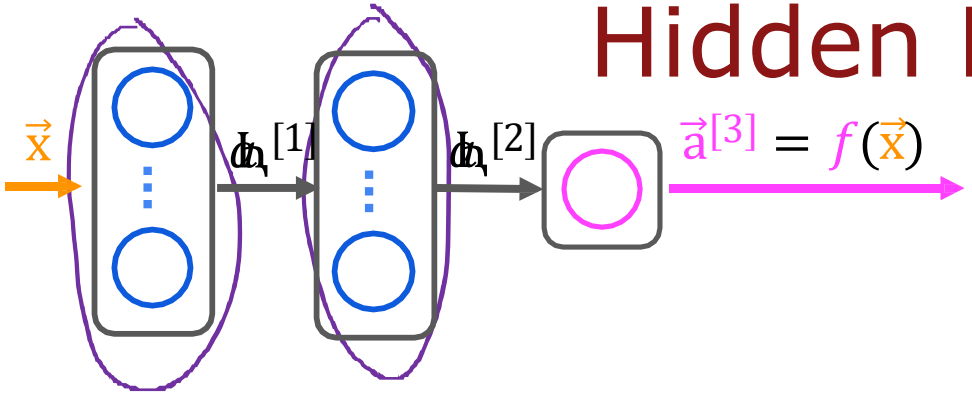
Regression

ReLU

Y = 0 or +

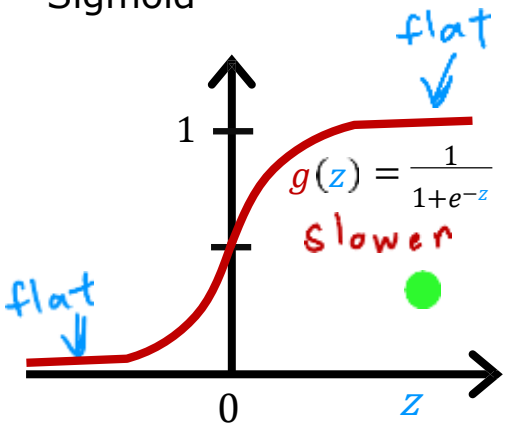


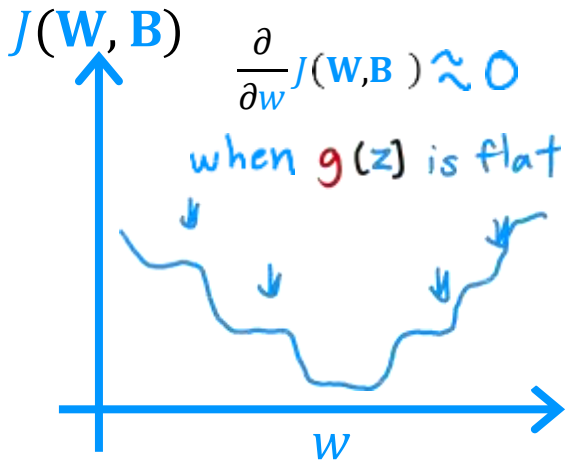
Hidden Layer



Choosing $g(z)$ for hidden layer

Sigmoid

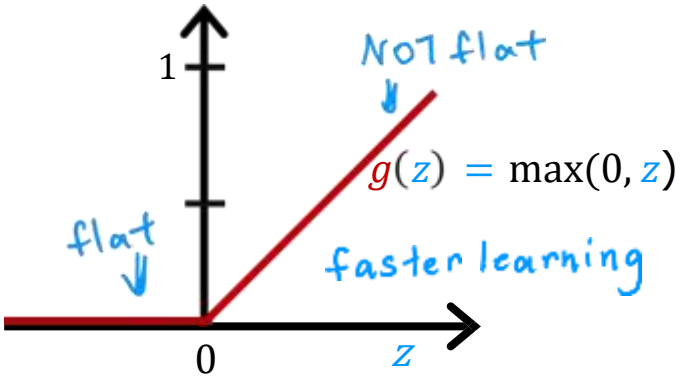




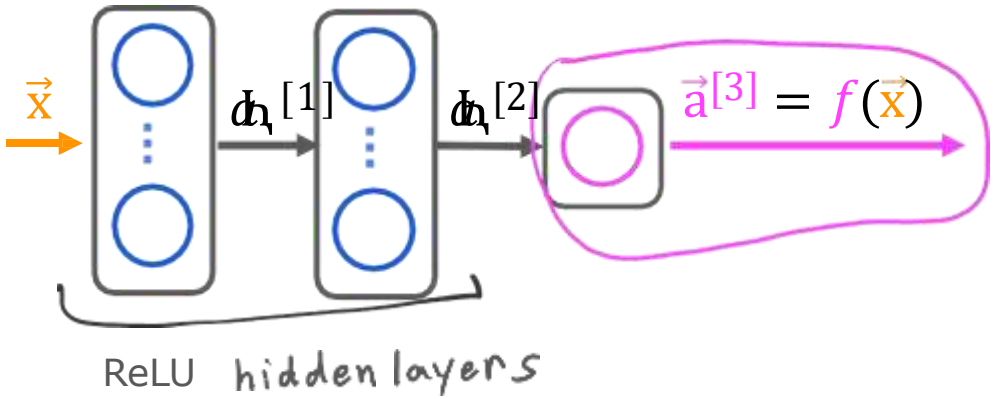
most common choice

ReLU

faster



Choosing Activation Summary



binary classification
activation='sigmoid'

regression y negative/positive
activation='linear'

regression $y \geq 0$
activation='relu'

```
from tf.keras.layers import Dense
model = Sequential([
  Dense(units=25, activation='relu'),
  Dense(units=15, activation='relu'),
  Dense(units=1,
        activation='sigmoid'
  )
])
```

layer1
layer2
layer3

or 'linear'
or 'relu'

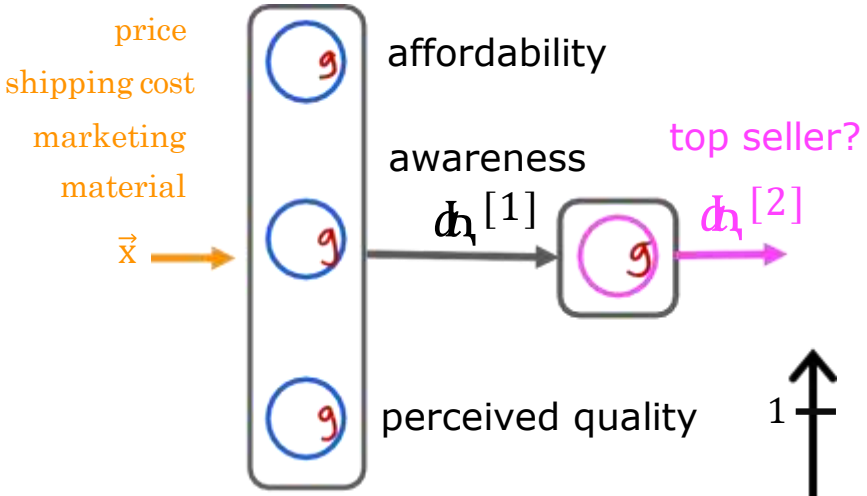
DeepLearning.AI

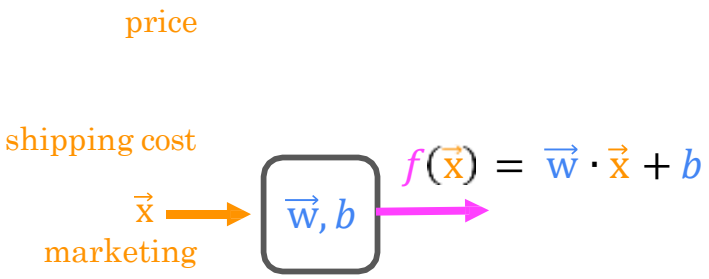
Stanford
ONLINE

Activation Functions

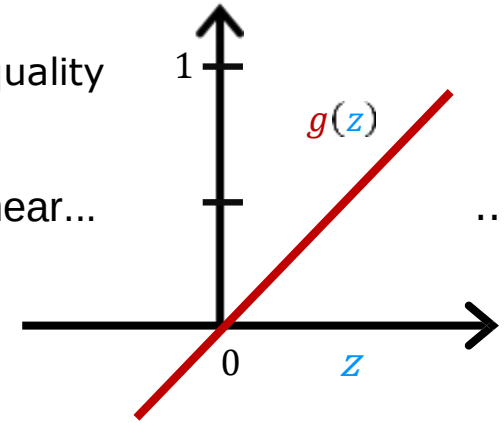
Why do we need activation functions?

Why do we need activation functions?



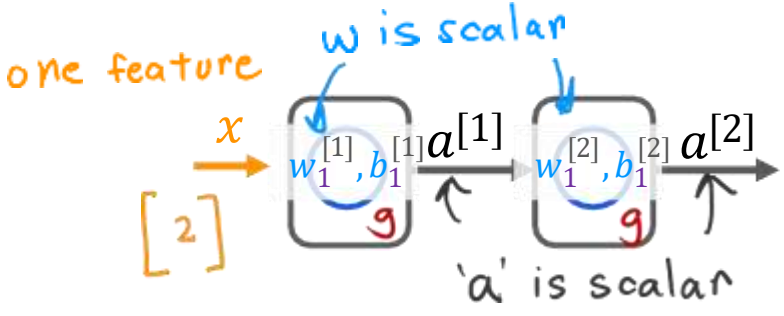


if all $g(z)$ are linear...



...no different than linear regression

Linear Example



$$g(z) = z$$

$$a^{[1]} = w_1^{[1]} x + b_1^{[1]}$$

$$a^{[2]} = w_1^{[2]} a^{[1]} + b_1^{[2]}$$

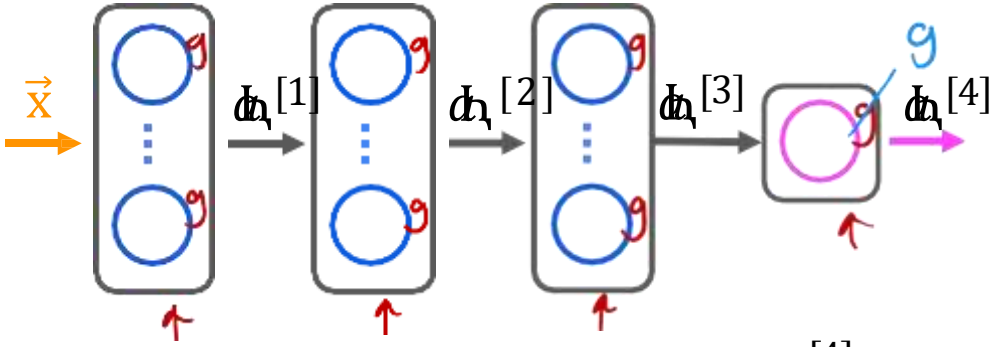
$$= w_1^{[2]} (w_1^{[1]} x + b_1^{[1]}) + b_1^{[2]}$$

$$a^{[2]} = (\underbrace{w_1^{[2]} w_1^{[1]}}_{w}) x + \underbrace{w_1^{[2]} b_1^{[1]} + b_1^{[2]}}_{b}$$

$$a^{[2]} = w x + b$$

$f(x) = wx + b$ linear regression

Example



$g(z) = z$

$$a_1^{[4]} = \vec{w}_1^{[4]} \cdot a_1^{[3]} + b_1^{[4]}$$

all linear (including output)
↳ equivalent to linear regression

$$a_1^{[4]} = \frac{1}{1+e^{-(\vec{w}_1^{[4]} \cdot \vec{a}^{[3]} + b_1^{[4]})}}$$

output activation is sigmoid
(hidden layers still linear)
↳ equivalent to logistic regression

Don't use linear activations in hidden layers (use ReLU)

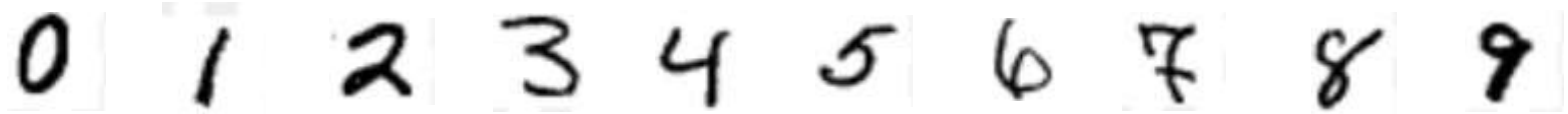
DeepLearning.AI

Stanford
ONLINE

Multiclass Classification

Multiclass

MNIST example

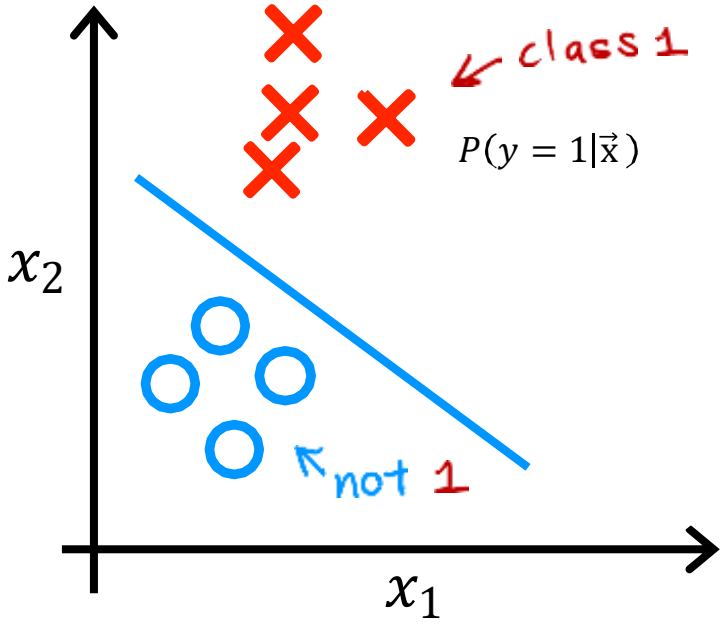


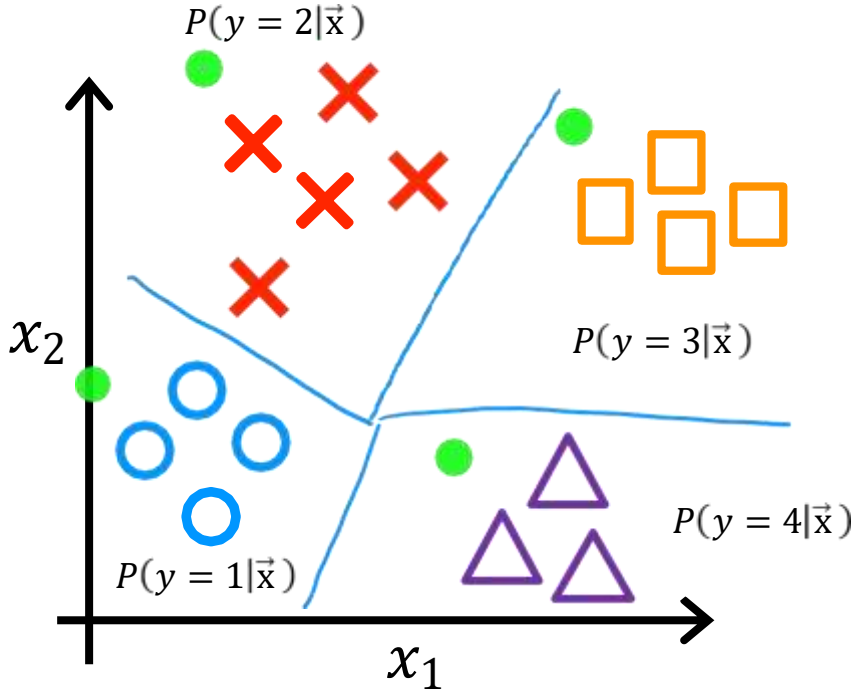
$y = 0$ 1 2 3 4 5 6 7 8 9

$y = 7$

multiclass classification problem:
target y can take on more than two possible values

Multiclass classification example





DeepLearning.AI

Stanford
ONLINE

Multiclass Classification

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 $y = 0$

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

Softmax regression

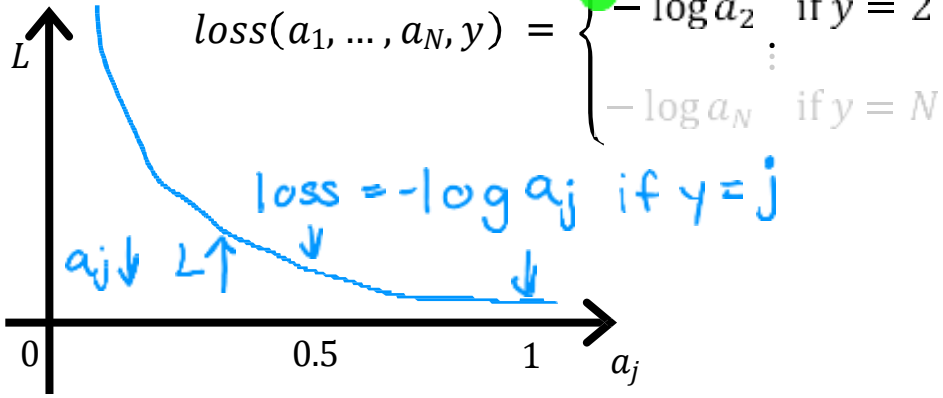
$$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + \cdots + e^{z_N}} = P(y = 1|\vec{x})$$

$$\vdots$$

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

Crossentropy loss

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



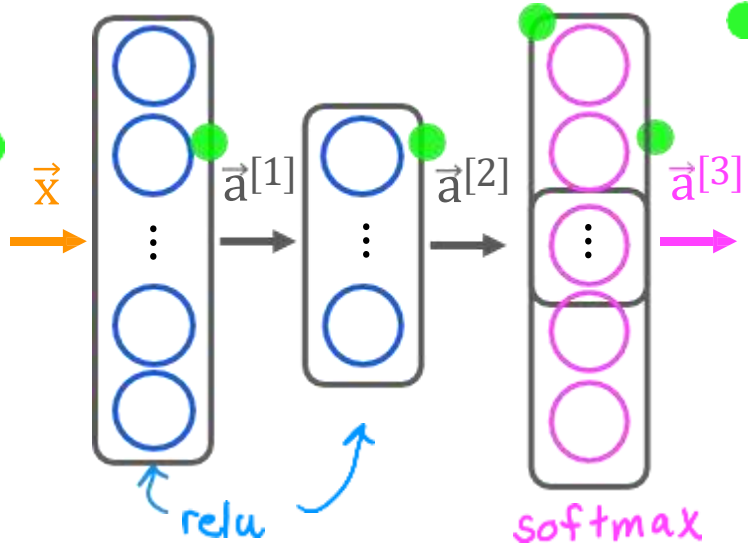
DeepLearning.AI

Stanford
ONLINE

Multiclass Classification

Neural Network with Softmax output

Neural Network with Softmax output



$$z_1 = \vec{w}_1 \cdot \vec{a}^{[2]} + b_1 \qquad a_1 = \frac{e^{z_1}}{e^{z_1} + \cdots + e^{z_{10}}} = P(y = 1 | \vec{x})$$

$$\vdots$$

$$z_{10} = \vec{w}_{10} \cdot \vec{a}^{[2]} + b_{10} \qquad a_{10} = \frac{e^{z_{10}}}{e^{z_1} + \cdots + e^{z_{10}}} = P(y = 10 | \vec{x})$$

logistic regression

$$a_1^{[3]} = g\left(z_1^{[3]}\right) \quad a_2^{[3]} = g\left(z_2^{[3]}\right)$$

softmax

$$\vec{a}^{[3]} = \left(a_1^{[3]}, \ldots a_{10}^{[3]}\right) = g\left(z_1^{[3]}, \ldots, z_{10}^{[3]}\right)$$

MNIST with softmax

1. specify the model

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

2. specify loss and cost

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

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!

DeepLearning.AI

Stanford ONLINE

Multiclass Classification

Improved implementation of softmax

Numerical Roundoff Errors

option 1

$$x = \frac{2}{10{,}000}$$

option 2

$$x = \left(1 + \frac{1}{10{,}000}\right) - \left(1 - \frac{1}{10{,}000}\right) =$$

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) )
```

'linear'

logit: z

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')
])
```

'linear'

```
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=SparseCrossEntropy(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 is $a_1 \ldots a_{10}$ $z_1 \ldots z_{10}$

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

fit

```
logit = model(X)
```

predict

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

DeepLearning.AI

Stanford
ONLINE

Multi-label Classification

Classification with multiple outputs (Optional)

Multi-label Classification

x

Is there a car?
Is there a bus?
Is there a pedestrian

yes
no
yes

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

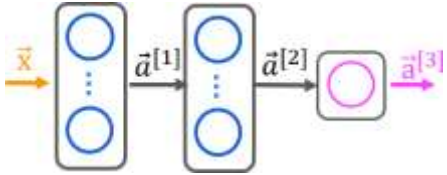
no
no
yes

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

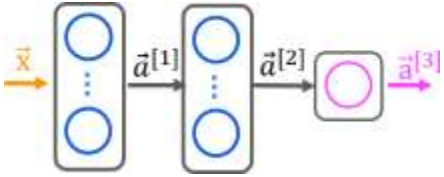
yes
yes
no

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

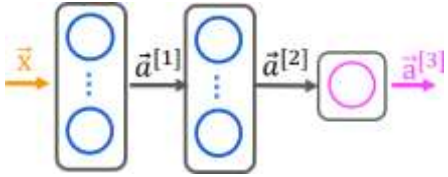
Multiple classes



car

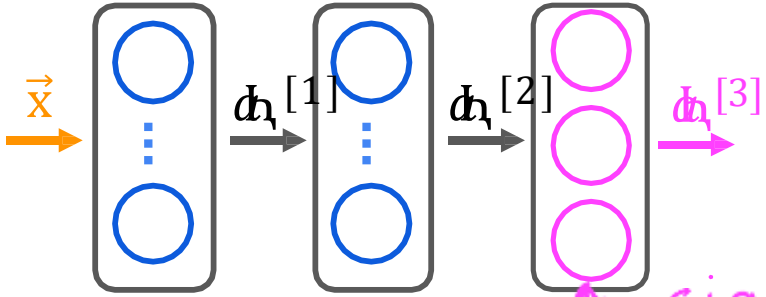


bus



pedestrian

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

DeepLearning.AI

Stanford
ONLINE

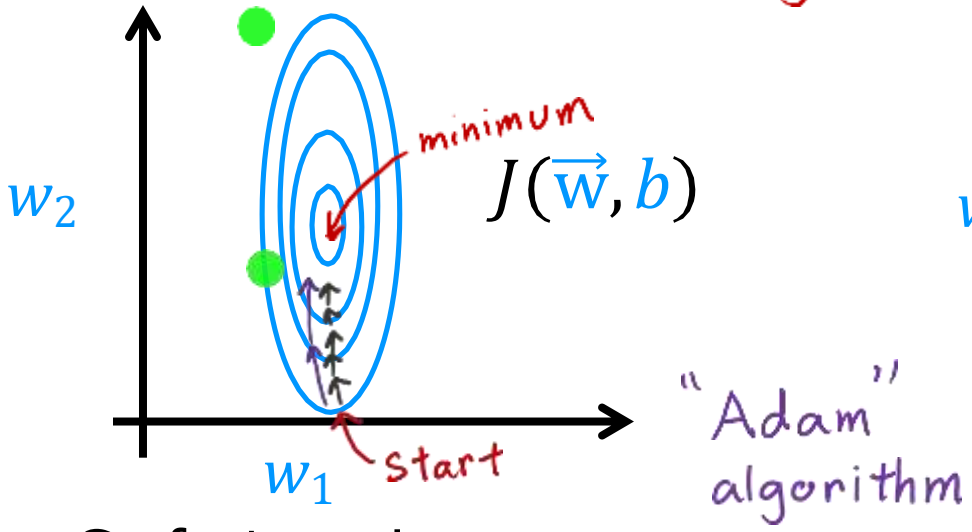
Additional Neural Network Concepts

Advanced Optimization

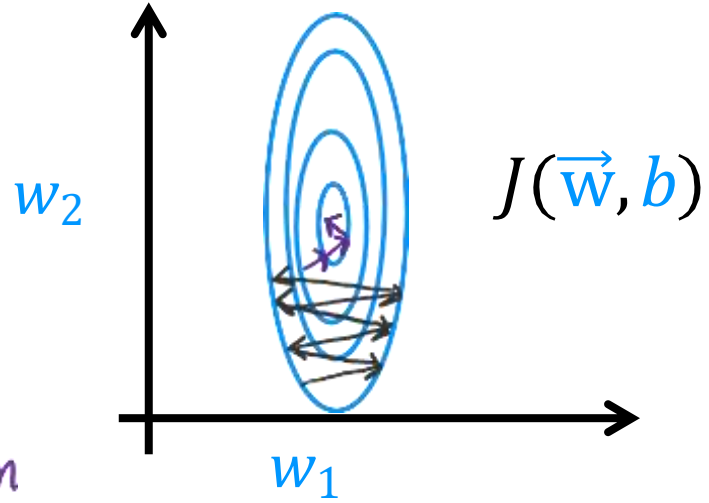
Gradient Descent

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

learning rate



Go faster – increase α



Go slower – decrease α

Adam Algorithm Intuition

Adam: Adaptive Moment estimation

not just one α

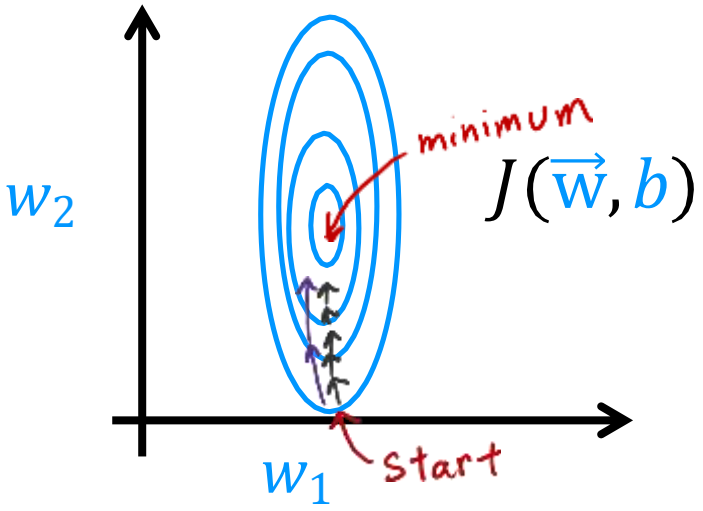
$$w_1 = w_1 - \alpha_1 \frac{\partial}{\partial w_1} J(\vec{w}, b)$$

$$\vdots$$

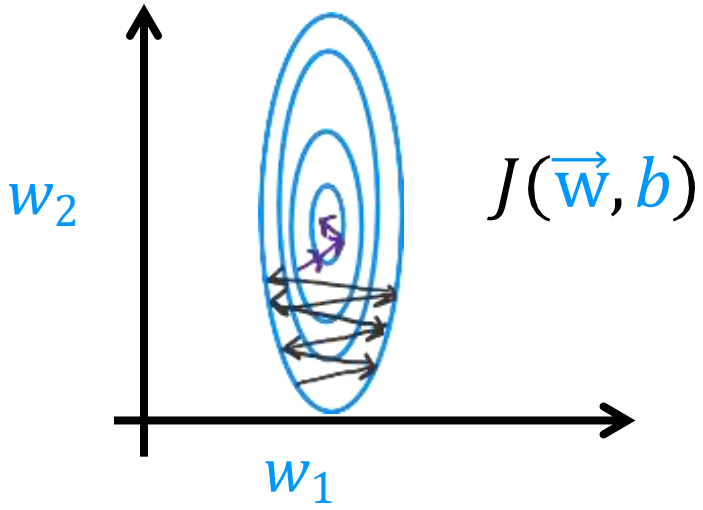
$$w_{10} = w_{10} - \alpha_{10} \frac{\partial}{\partial w_{10}} J(\vec{w}, b)$$

$$b = b - \alpha_{11} \frac{\partial}{\partial b} J(\vec{w}, b)$$

Adam Algorithm Intuition



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)
```

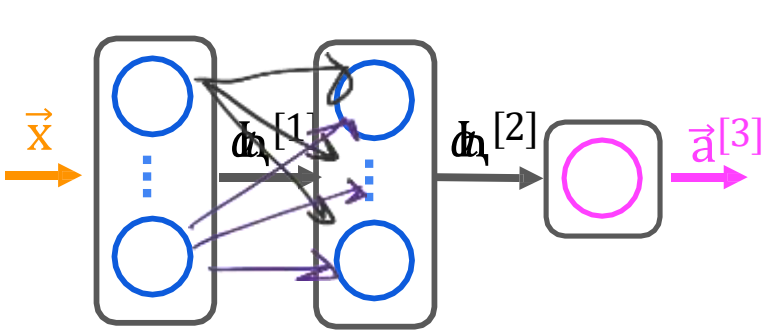
DeepLearning.AI

Stanford
ONLINE

Additional Neural Network Concepts

Additional Layer Types

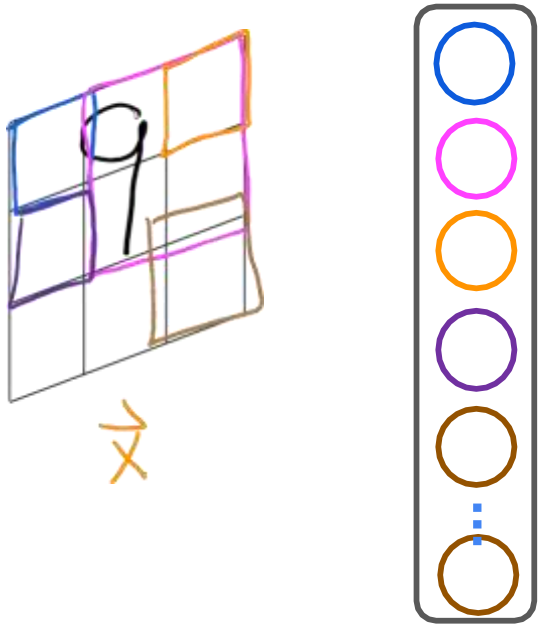
Dense Layer



Each neuron output is a function of all the activation outputs of the previous layer.

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

Convolutional Layer

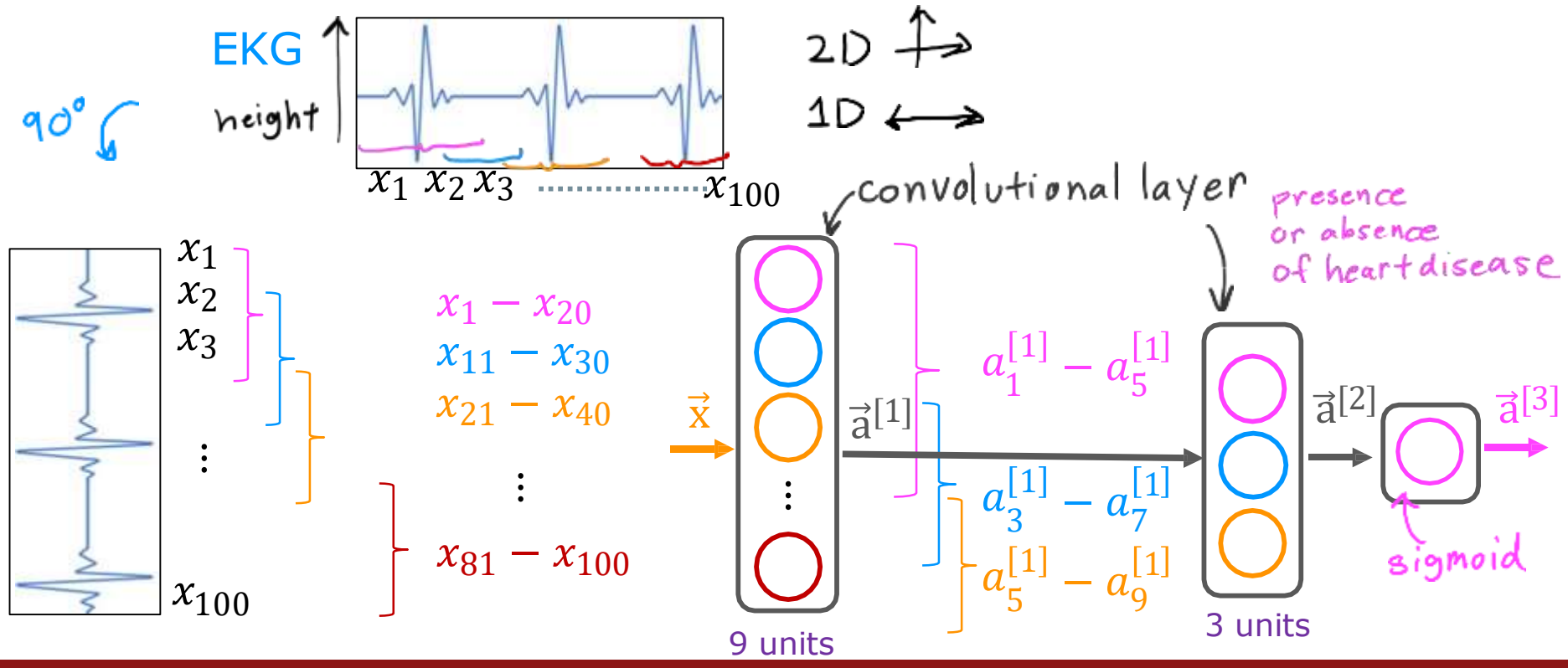


Each Neuron only looks at part of the previous layer's inputs.

Why?

- Faster computation
- Need less training data (less prone to overfitting)

Convolutional Neural Network



Copyright Notice

DeepLearning.AI

Stanford
ONLINE

Advice for applying machine learning

Deciding what to try next

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
- Try smaller sets of features
- Try getting additional features
- Try adding polynomial features $(x_1^2, x_2^2, x_1 x_2, etc)$
- Try decreasing λ
- Try increasing λ

Machine learning diagnostic

Diagnostic: A test that you run to gain insight into what is/isn't working with a learning algorithm, to gain guidance into improving its performance.

Diagnostics can take time to implement but doing so can be a very good use of your time.

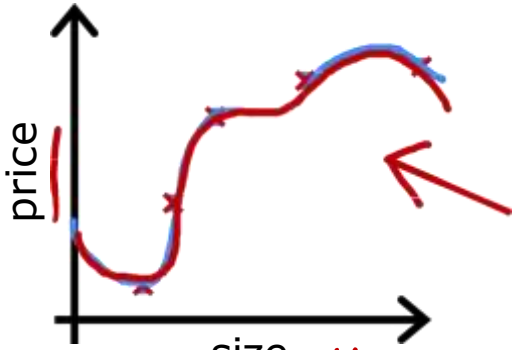
DeepLearning.AI

Stanford
ONLINE

Evaluating and
choosing models

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.

$x = x_1$

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

$w_n x^4$

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

$f(x_1)$

Evaluating your model

Dataset:

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

70%

30%

training set

test set

$(x^{(1)}, y^{(1)})$
$(x^{(2)}, y^{(2)})$
$\vdots$
$(x^{(m_{train})}, y^{m_{train}})$

m_{train} = no. training examples = 7

$(x_{test}^{(1)}, y_{test}^{(1)})$
$\vdots$
$(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

m_{test} = no. test examples = 3

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

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

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

Compute test error:

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

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)

Fit parameters

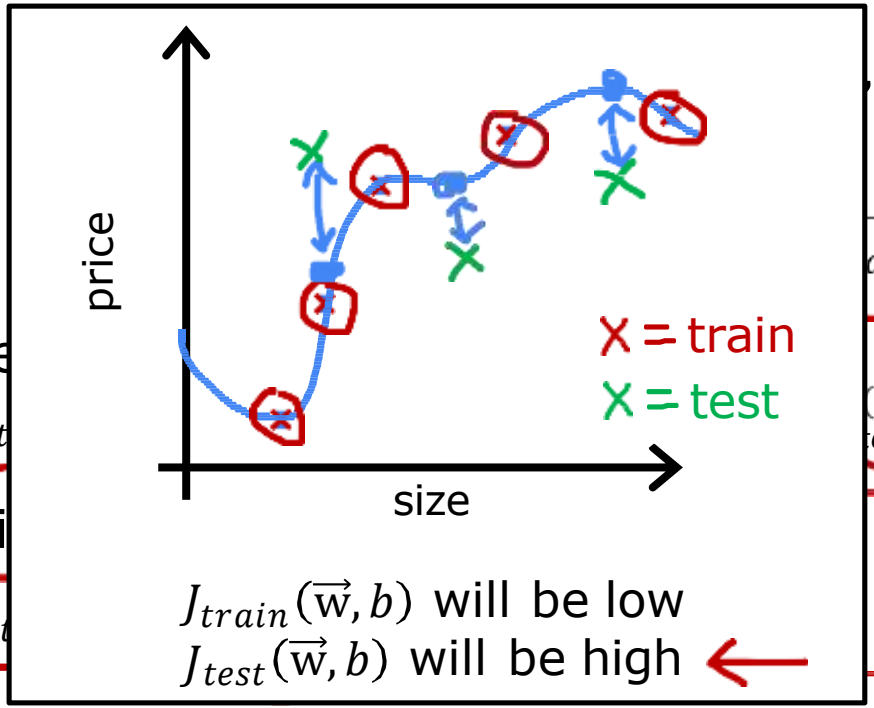
$J(\vec{w}, b) = \min_{\vec{w}, b}$

Compute test e

J_t

Compute traini

J_t



$, b)$

$\left[\frac{}{ain} \sum_{j=1}^{n} w_j^2 \right]$

$\left(\quad ^{(i)}_{est} \right)^2 \Big]$

$+$

$\left. - y^{(i)}_{train} \right)^2 \Big]$

Train/test procedure for classification problem

0/1

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

E.g.,

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

Compute test error:

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

Compute train error:

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

Train/test procedure for classification problem

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

E.g.,

$$J(\vec{w}, b) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log f_{\vec{w},b}(\right.$$

Compute test error:

$$J_{test}(\vec{w}, b) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} \left[y \right.$$

Compute train error:

$$J_{train}(\vec{w}, b) = -\frac{1}{m_{train}} \sum_{i=1}^{m_{trai}}$$

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

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

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.

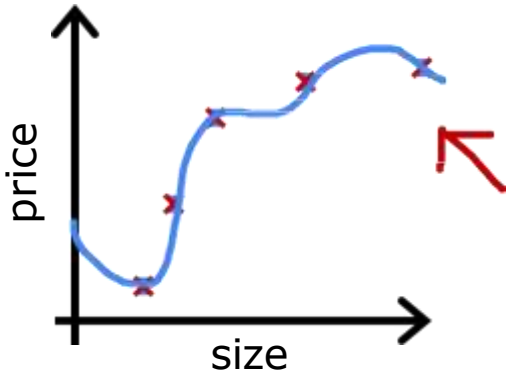
DeepLearning.AI

Stanford
ONLINE

Evaluating and choosing models

Model selection and training/cross validation/test sets

Model selection (choosing a model)



$$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x^2 + w_3x^3 + w_4x^4 + b$$

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 than $J_{train}(\vec{w}, b)$.

Model selection (choosing a model)

d=1 1. $f_{\vec{w},b}(\vec{x}) = w_1 x_1 + b$ → $w^{<1>}, b^{<1>}$ → $J_{test}(w^{<1>}, b^{<1>})$

d=2 2. $f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x^2 + b$ → $w^{<2>}, b^{<2>}$ → $J_{test}(w^{<2>}, b^{<2>})$

d=3 3. $f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x^2 + w_3 x^3 + b$ → $w^{<3>}, b^{<3>}$ → $J_{test}(w^{<3>}, b^{<3>})$

$\vdots$

d=10 10. $f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x^2 + \cdots + w_{10} x^{10} + b$ → $J_{test}(w^{<10>}, b^{<10>})$

Choose $w_1 x_1 + \cdots + w_5 x^5 + b$ 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 is $J_{test}(w^{<5>}, b^{<5>})$ is likely to be an optimistic estimate of generalization error. Ie: An extra parameter d (degree of polynomial) was chosen using the test set.

w,b

Training/cross validation/test set

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

validation set
development set
dev set

training set
60%

cross validation
20%

test set
20%

$(x^{(1)}, y^{(1)})$
$\vdots$
$(x^{(m_{train})}, y^{(m_{train})})$
$m_{train} = 6$

$(x_{cv}^{(1)}, y_{cv}^{(1)})$
$\vdots$
$(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$
$m_{cv} = 2$

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

Training/cross validation/test set

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

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

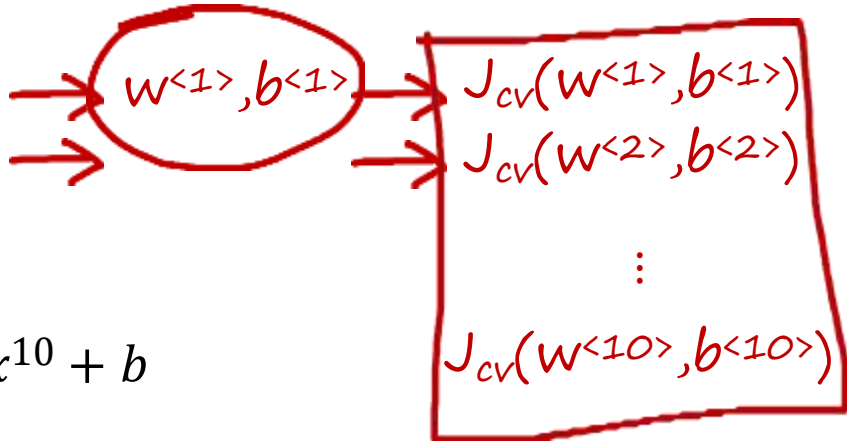
d=1 1. $f_{\vec{w},b}(\vec{x}) = w_1 x_1 + b$

d=2 2. $f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x^2 + b$

d=3 3. $f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x^2 + w_3 x^3 + b$

⋮ ⋮

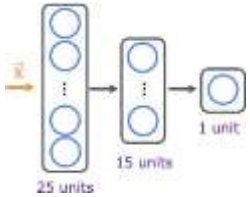
d=10 10. $f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x^2 + \cdots + w_{10} x^{10} + b$



→ Pick $w_1 x_1 + \cdots + w_4 x^4 + b$ (J_{cv} $w^{<4>}, b^{<4>}$)

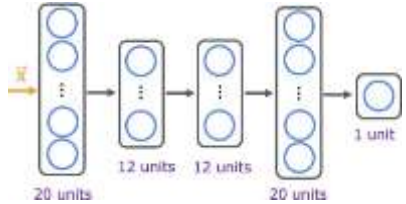
Estimate generalization error using test the set: $J_{test}\ (w^{<4>}, b^{<4>})$

Model selection – choosing a neural network architecture

1. 

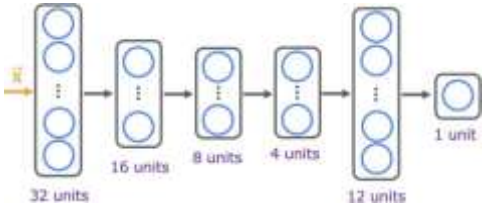
$w^{(1)}, b^{(1)}$

$J_{cv}(\mathbf{W}^{(1)}, \mathbf{B}^{(1)})$

2. 

$w^{(2)}, b^{(2)}$

$J_{cv}(\mathbf{W}^{(2)}, \mathbf{B}^{(2)})$

3. 

$w^{(3)}, b^{(3)}$

$J_{cv}(\mathbf{W}^{(3)}, \mathbf{B}^{(3)})$

Train, CV

Pick $\mathbf{W}^{(2)}, \mathbf{B}^{(2)}$

Estimate generalization error using the test set: $J_{test}(\mathbf{W}^{(2)}, \mathbf{B}^{(2)})$

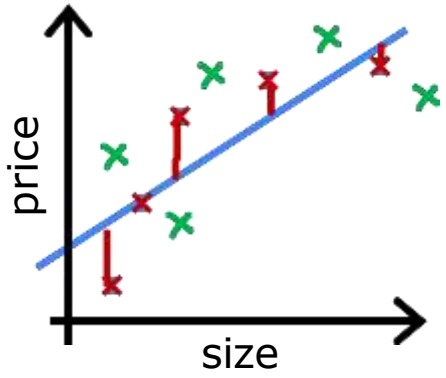
DeepLearning.AI

Stanford
ONLINE

Bias and variance

Diagnosing bias and variance

Bias/variance



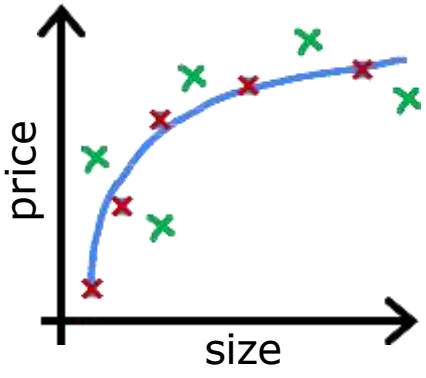
$f_{\vec{w},b}(x) = w_1 x + b$

High bias (underfit)

$d = 1$

J_{train} is high

J_{cv} is high



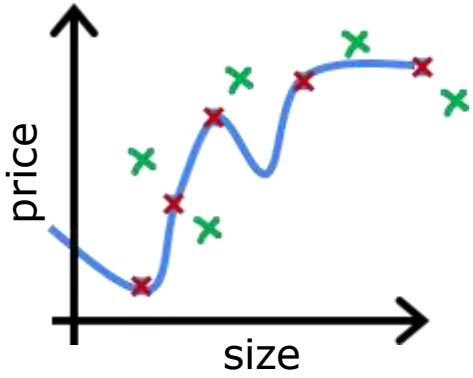
$f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + b$

"Just right"

$d = 2$

J_{train} is low

J_{cv} is low



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

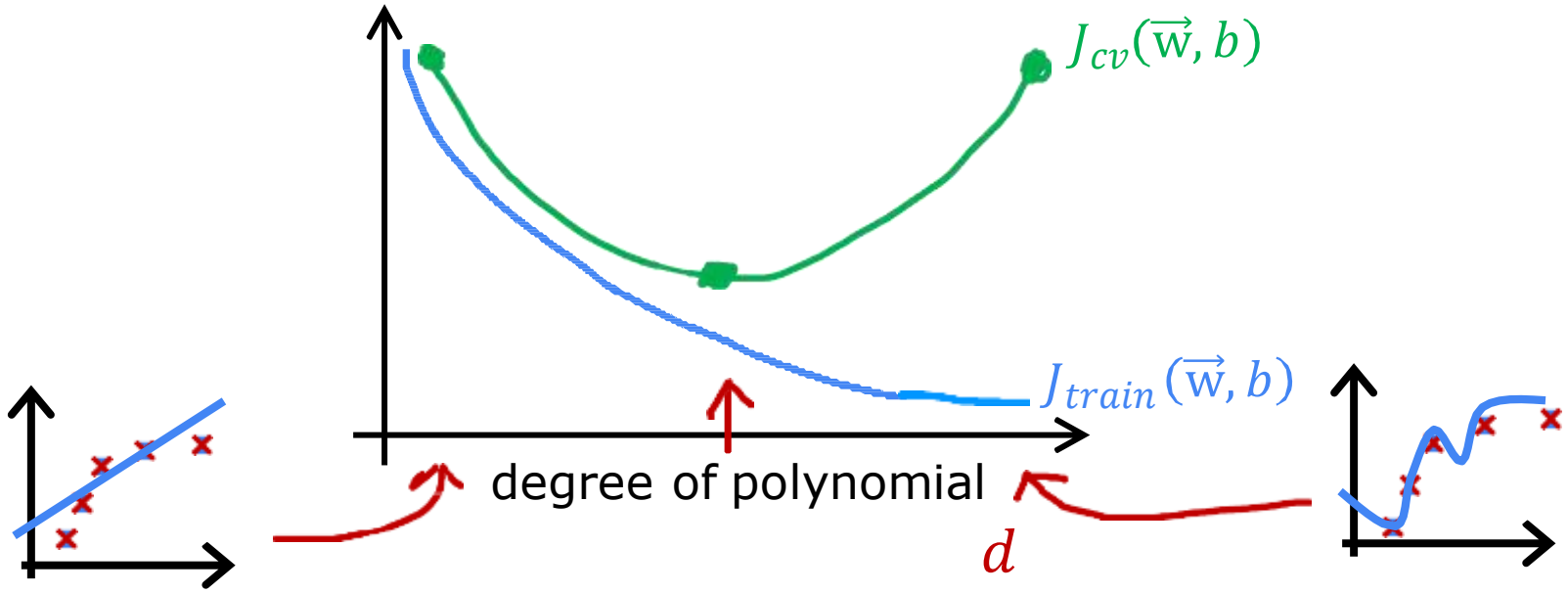
High variance (overfit)

$d = 4$

J_{train} is low

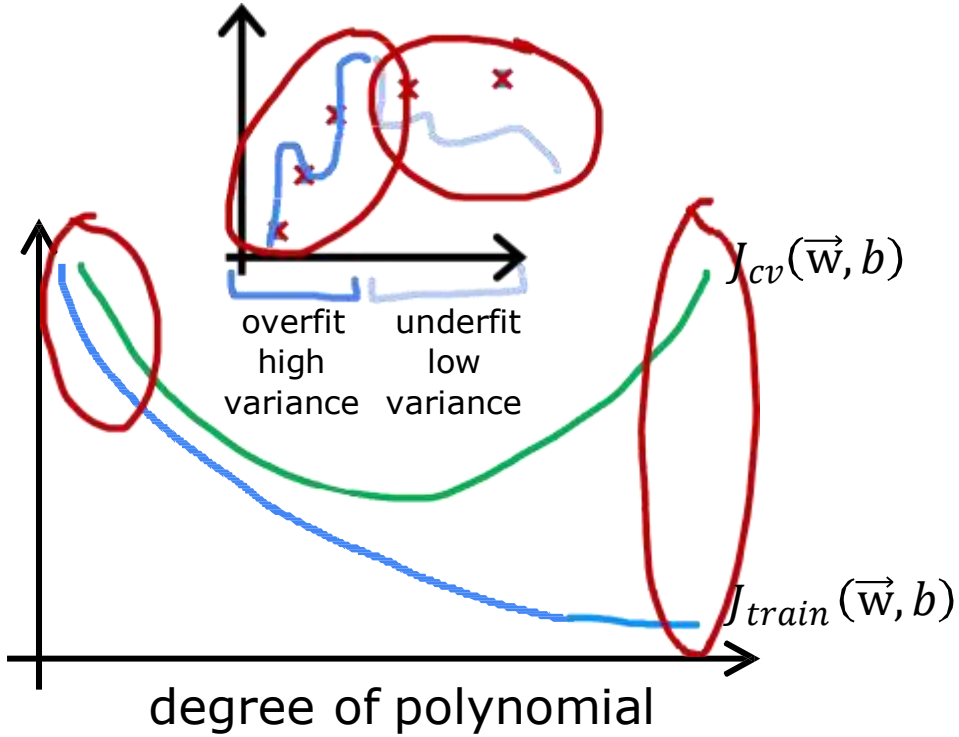
J_{cv} is high

Understanding bias and variance



Diagnosing bias and variance

How do you tell if your algorithm has a bias or variance problem?



High bias (underfit)

J_{train} will be high

($J_{train} \approx J_{cv}$)

High variance (overfit)

$J_{cv} \gg J_{train}$

(J_{train} may be be low)

High bias and high variance

J_{train} will be high

and $J_{cv} \gg J_{train}$

DeepLearning.AI

Stanford
ONLINE

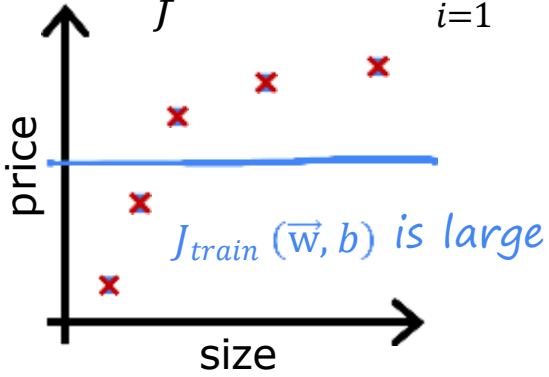
Bias and variance

Regularization and bias/variance

Linear regression with regularization

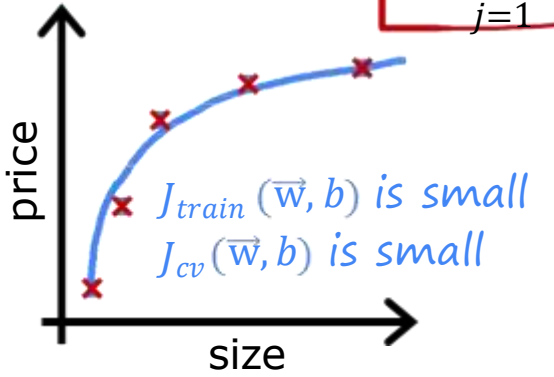
Model: $f_{\vec{w},b}(x) = w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$

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



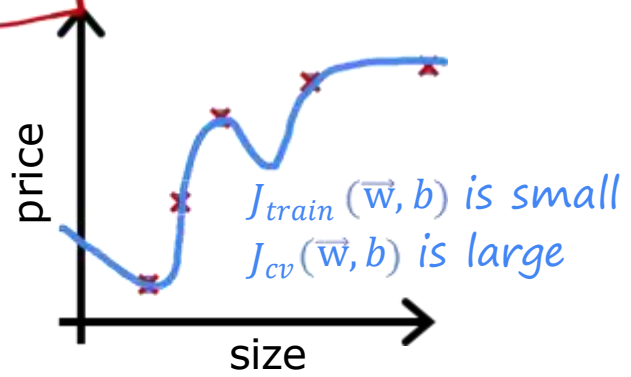
Large λ
High bias (underfit)

$\lambda = 10{,}000$ $w_1 \approx 0, w_2 \approx 0$
$f_{\vec{w},b}(\vec{x}) \approx b$



Intermediate λ

λ



Small λ
High variance (overfit)

$\lambda = 0$

Choosing the regularization parameter λ

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

1. Try $\lambda = 0$ → $\min_{\vec{w},b} J(\vec{w},b)$ → $w^{<1>}, b^{<1>}$ → $J_{cv}(w^{<1>}, b^{<1>})$
2. Try $\lambda = 0.01$ → $w^{<2>}, b^{<2>}$ → $J_{cv}(w^{<2>}, b^{<2>})$
3. Try $\lambda = 0.02$ → $J_{cv}(w^{<3>}, b^{<3>})$
4. Try $\lambda = 0.04$
5. Try $\lambda = 0.08$ → $J_{cv}(w^{<5>}, b^{<5>})$

⋮

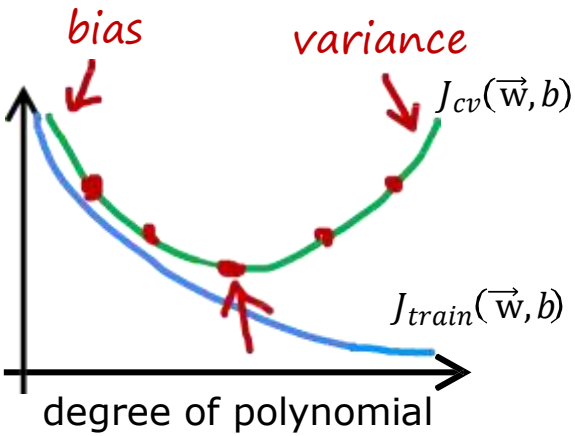
12. Try $\lambda \approx 10$ → $w^{<12>}, b^{<12>}$ → $J_{cv}(w^{<12>}, b^{<12>})$

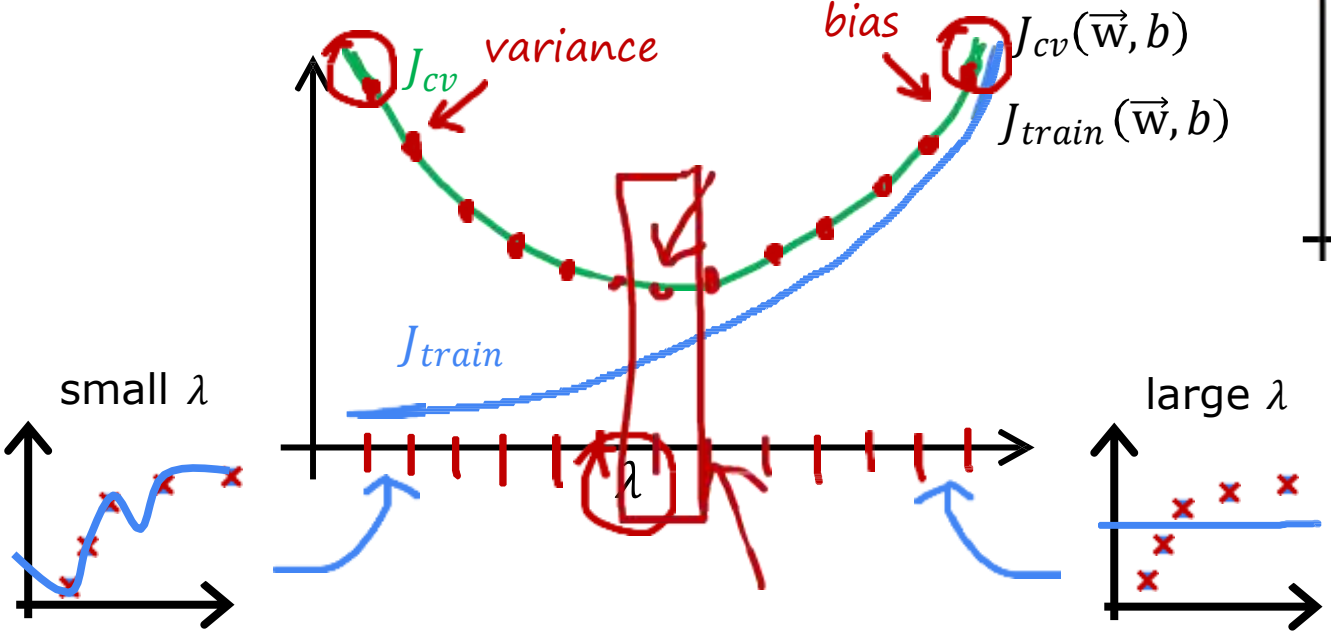
Pick $w^{<5>}, b^{<5>}$

Report test error: $J_{test}(w^{<5>}, b^{<5>})$

Bias and variance as a function of regularization parameter λ

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





DeepLearning.AI

Stanford
ONLINE

Bias and variance

Establishing a baseline level of performance

Speech recognition example

Human level performance : 10.6%

Training error J_{train} : 10.8%

Cross validation error J_{cv} : 14.8%

0.2%

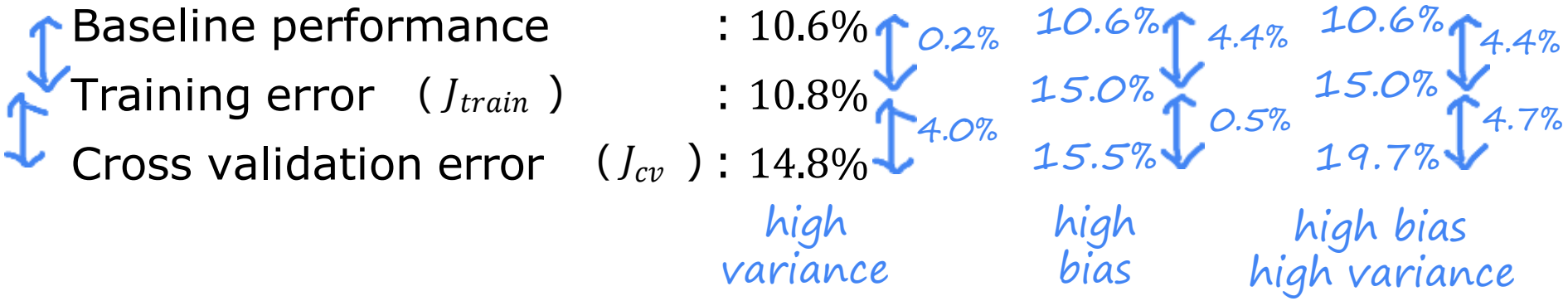
4.0%

Establishing a baseline level of performance

What is the level of error you can reasonably hope to get to?

- Human level performance
- Competing algorithms performance
- Guess based on experience

Bias/variance examples



DeepLearning.AI

Stanford
ONLINE

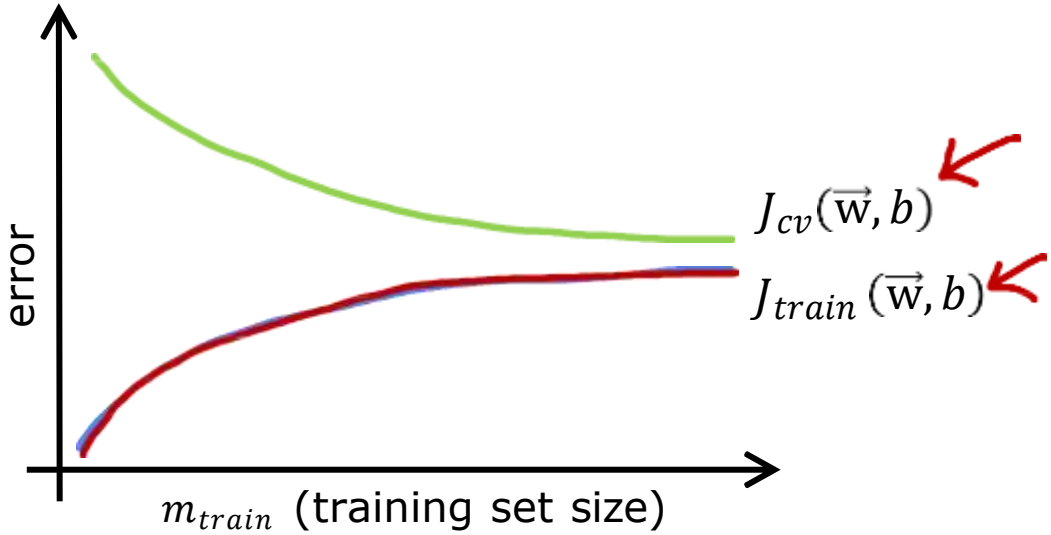
Bias and variance

Learning curves

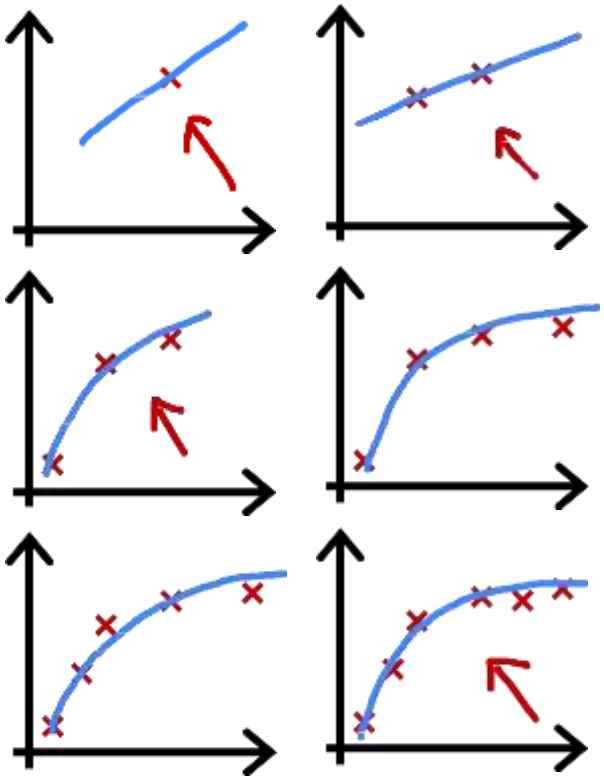
Learning curves

J_{train}= training error

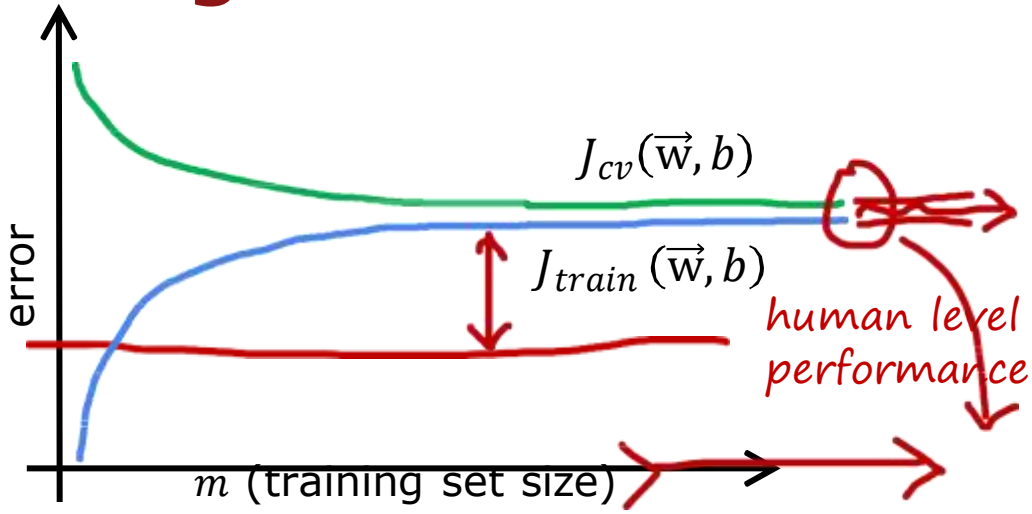
J_{cv}= cross validation error



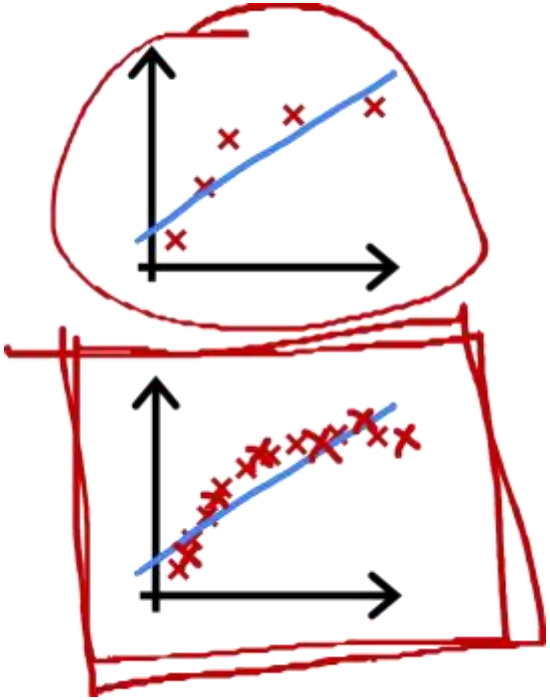
$f_{\vec{w},b}(x) = w_1x + w_2x^2 + b$



High bias

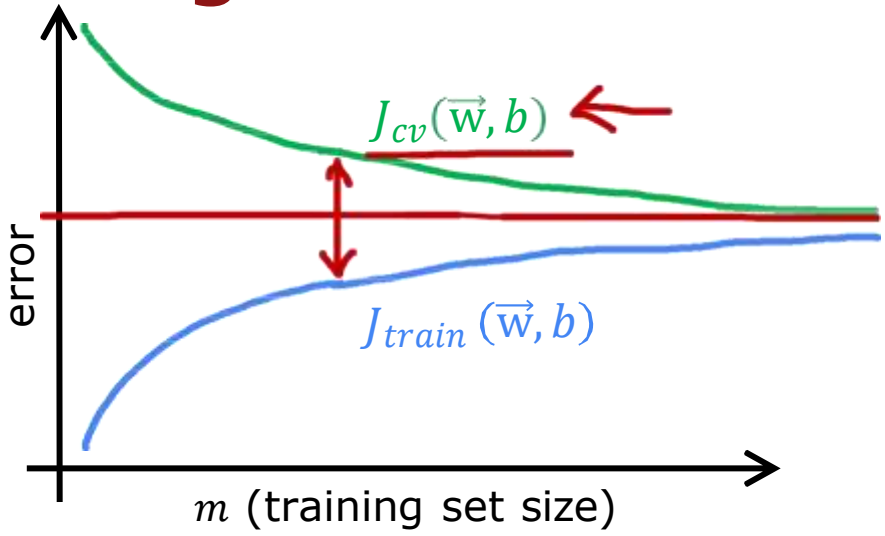


$f_{\vec{w},b}(x) = w_1 x + b$



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

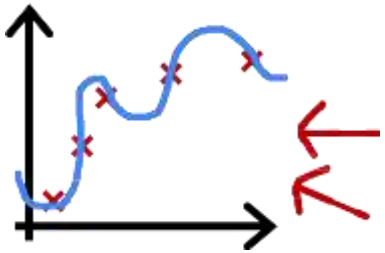
High variance

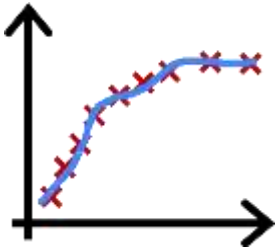


if a learning algorithm suffers from high variance, getting more training data is likely to help.

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

(with small λ)





DeepLearning.AI

Stanford
ONLINE

Bias and variance

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 $x, x^2, x^3, x^4, x^5 \ldots$ — fixes high variance
- Try getting additional features — fixes high bias
- Try adding polynomial features $(x_1^2, x_2^2, x_1 x_2, etc)$ — fixes high bias
- Try decreasing λ — fixes high bias
- Try increasing λ — fixes high variance

DeepLearning.AI

Stanford
ONLINE

Bias and variance

Bias/variance and neural networks

The bias variance tradeoff

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

Simple model

High bias

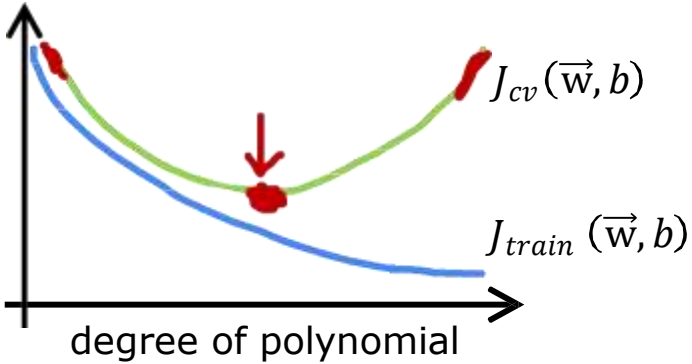
$f_{\vec{w},b}(x) = w_1x + w_2x^2 + b$

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

Complex model

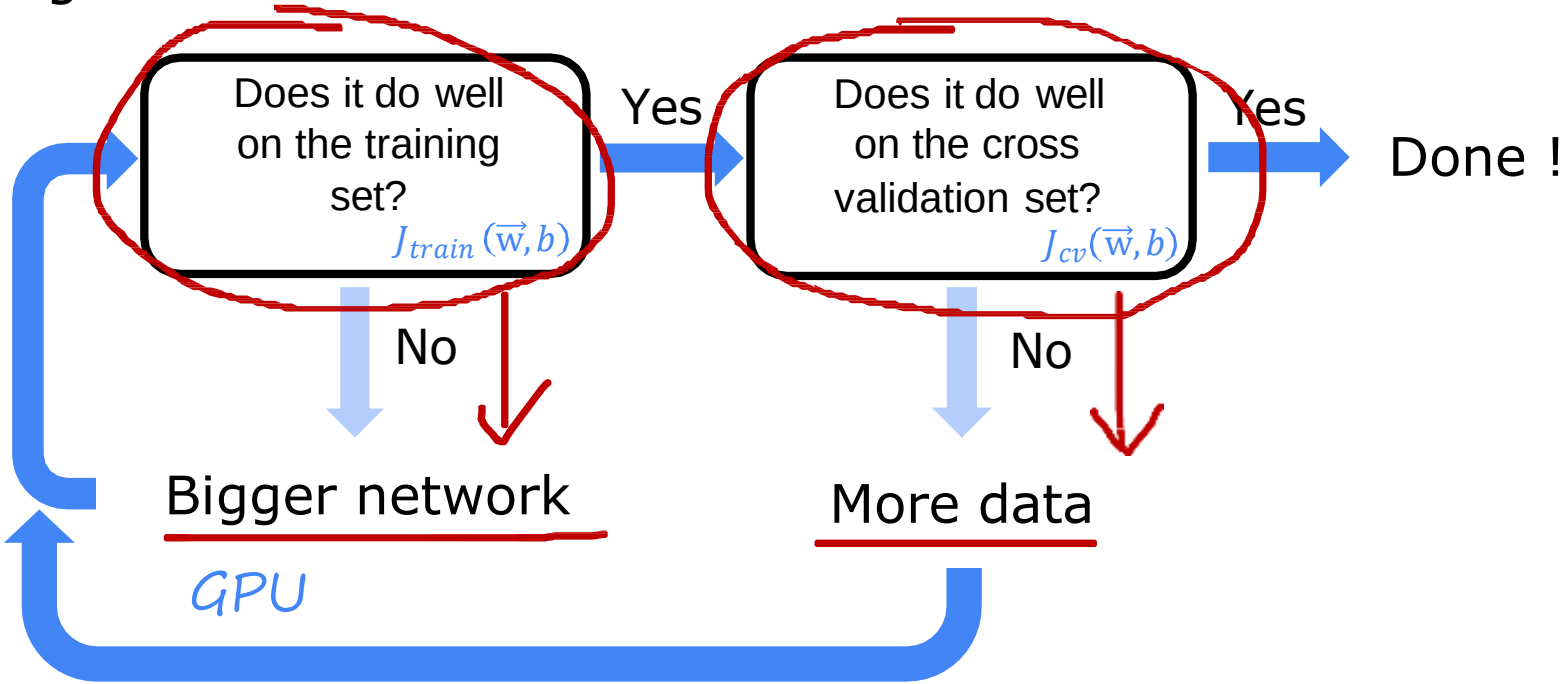
High variance

tradeoff

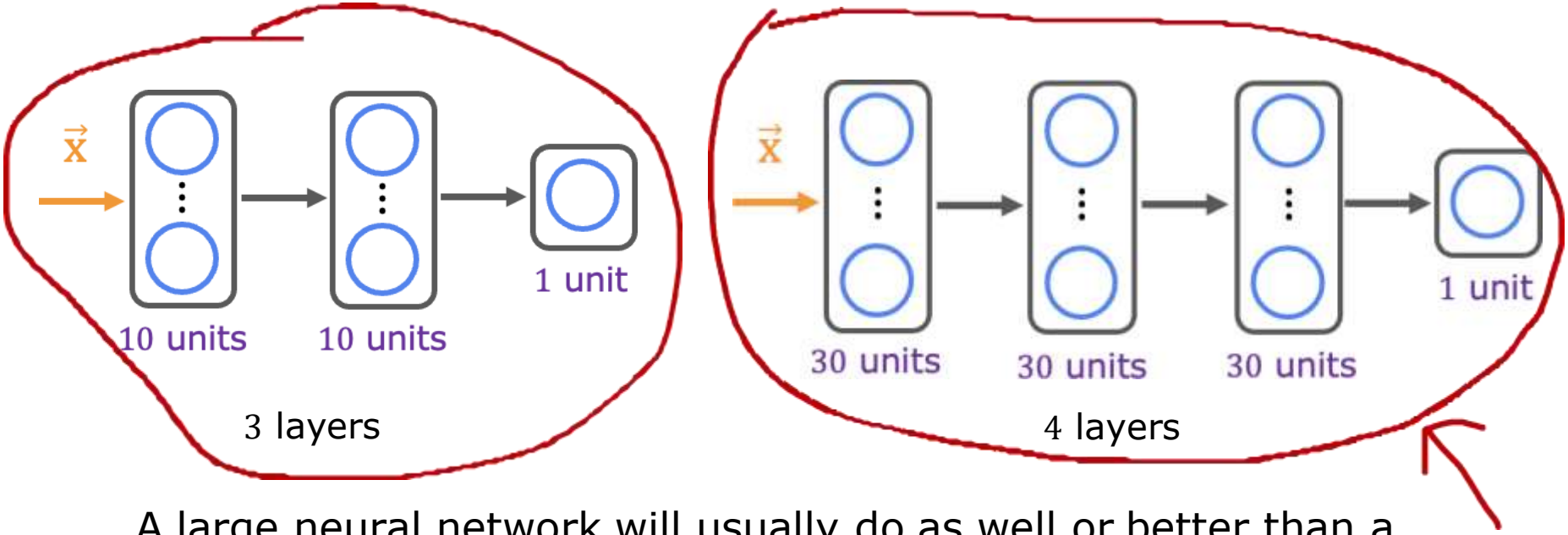


Neural networks and bias variance

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

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

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])
```

DeepLearning.AI

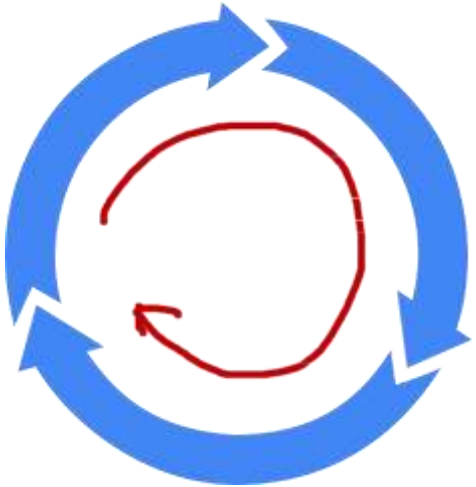
Stanford
ONLINE

Machine learning development process

Iterative loop of ML development

Iterative loop of ML development

Choose architecture
(model, data, etc.)



Train
model

Diagnostics
(bias, variance and
error analysis)

Spam classification example

From: cheapsales@buystufffromme.com
To: Andrew Ng
Subject: Buy now!

Deal of the week! Buy now!
Rolex w4tchs - $100
Med1cine (any kind) - £50
Also low cost M0rgages
available.

From: Alfred Ng
To: Andrew Ng
Subject: Christmas dates?

Hey Andrew,
Was talking to Mom about plans for Xmas. When do you get off work. Meet Dec 22?
Alf

Building a spam classifier

Supervised learning: $\vec{x}$ = features of email
y = spam (1) or not spam (0)

Features: list the top 10,000 words to compute $x_1, x_2, \cdots, x_{10,000}$

$$\vec{x} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ \vdots \end{bmatrix}$$

a
andrew
buy
deal
discount
⋮

From: cheapsales@buystufffromme.com
To: Andrew Ng
Subject: Buy now!

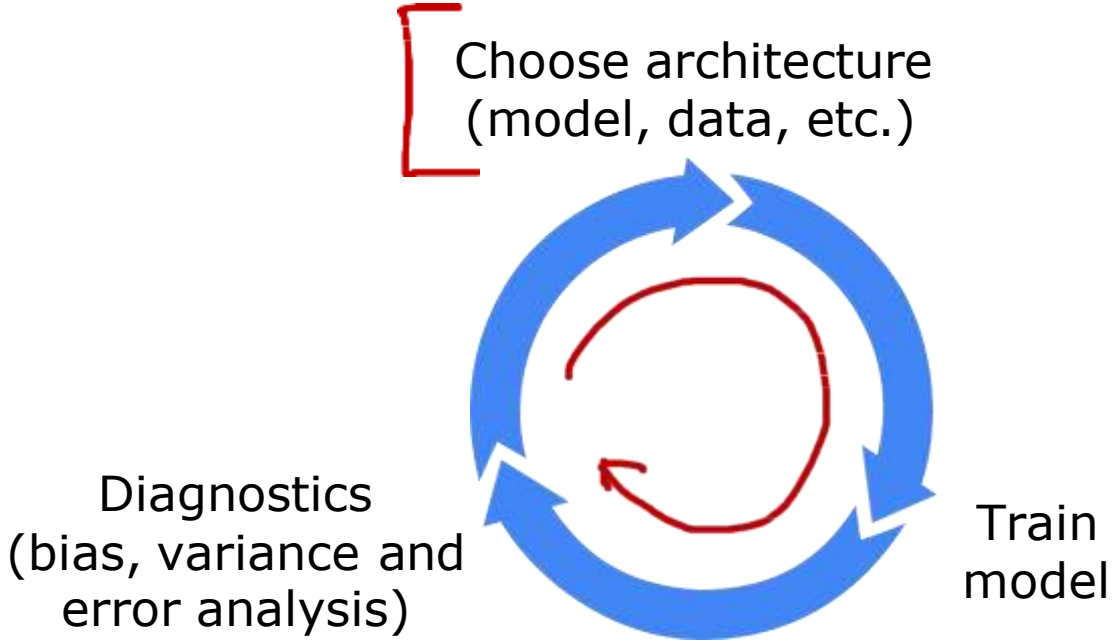
Deal of the week! Buy now!
Rolex w4tchs - $100
Med1cine (any kind) - £50
Also low cost M0rgages
available.

Building a spam classifier

How to try to reduce your spam classifier's error?

- Collect more data. E.g., "Honeypot" project.
- Develop sophisticated features based on email routing (from email header).
- Define sophisticated features from email body. E.g., should "discounting" and "discount" be treated as the same word.
- Design algorithms to detect misspellings. E.g., w4tches, med1cine, m0rtgage.

Iterative loop of ML development



DeepLearning.AI

Stanford
ONLINE

Machine learning development process

Error analysis

Error analysis

m_{cv}= ~~500~~ 5000 examples in cross validation set.

Algorithm misclassifies ~~100~~ 1000 of them.

Manually examine 100 examples and categorize them based on common traits.

- Pharma: 21 → more data features
- Deliberate misspellings (w4tches, med1cine): 3
- Unusual email routing: 7
- Steal passwords (phishing): 18 → more data features
- Spam message in embedded image: 5

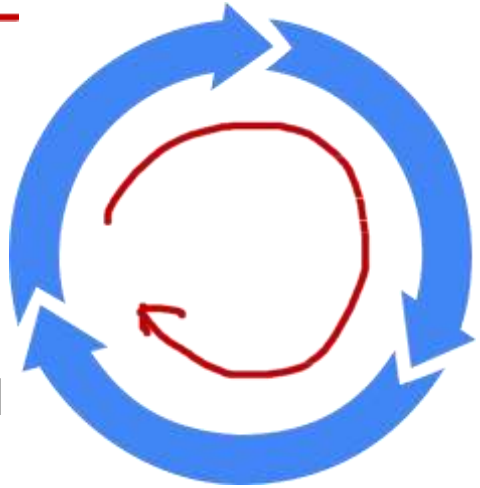
Building a spam classifier

How to try to reduce your spam classifier's error?

- Collect more data. E.g., "Honeypot" project.
- Develop sophisticated features based on email routing (from email header).
- Define sophisticated features from email body. E.g., should "discounting" and "discount" be treated as the same word.
- Design algorithms to detect misspellings. E.g., w4tches, med1cine, m0rtgage.

Iterative loop of ML development

Choose architecture
(model, data, etc.)



Train
model

Diagnostics
(bias, variance and
error analysis)

DeepLearning.AI

Stanford
ONLINE

Machine learning development process

Adding data

Adding data

- Add more data of everything. E.g., "Honeypot" project.
- Add more data of the types where error analysis has indicated it might help.

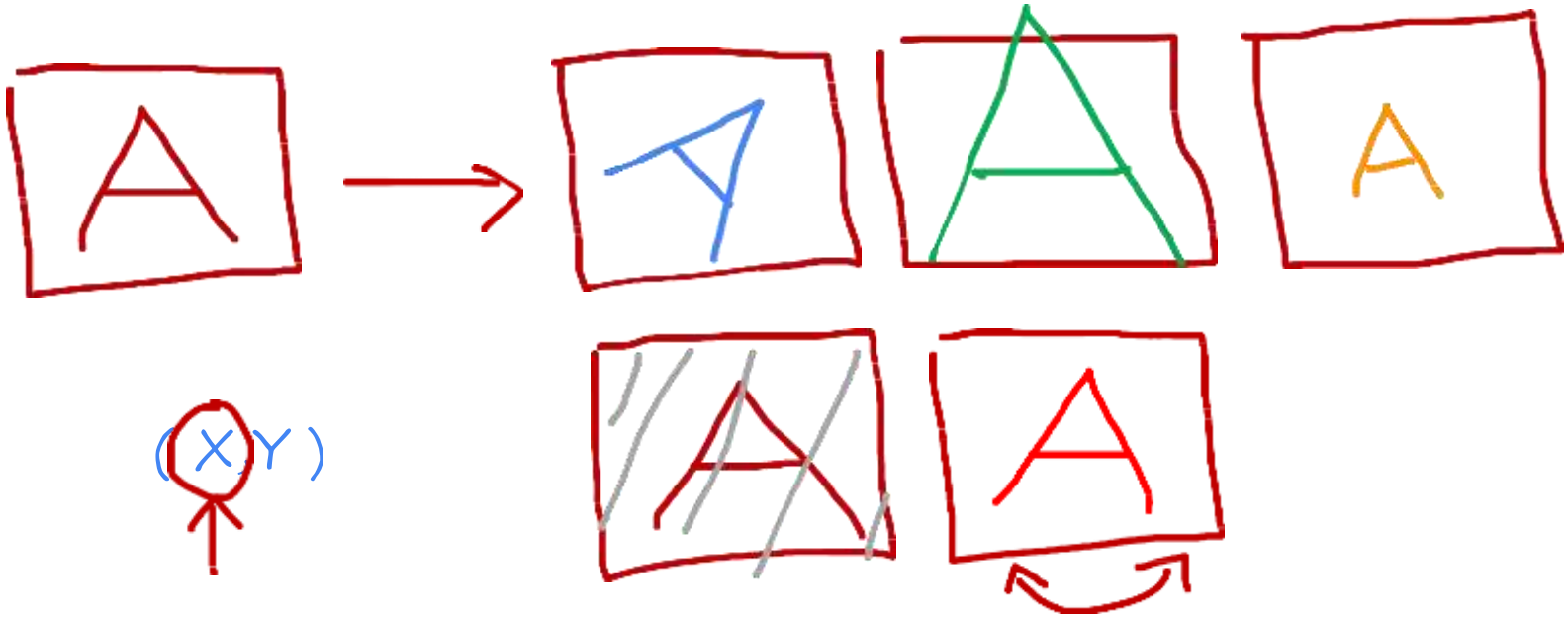
Pharma spam

E.g., Go to unlabeled data and find more examples of Pharma related spam.

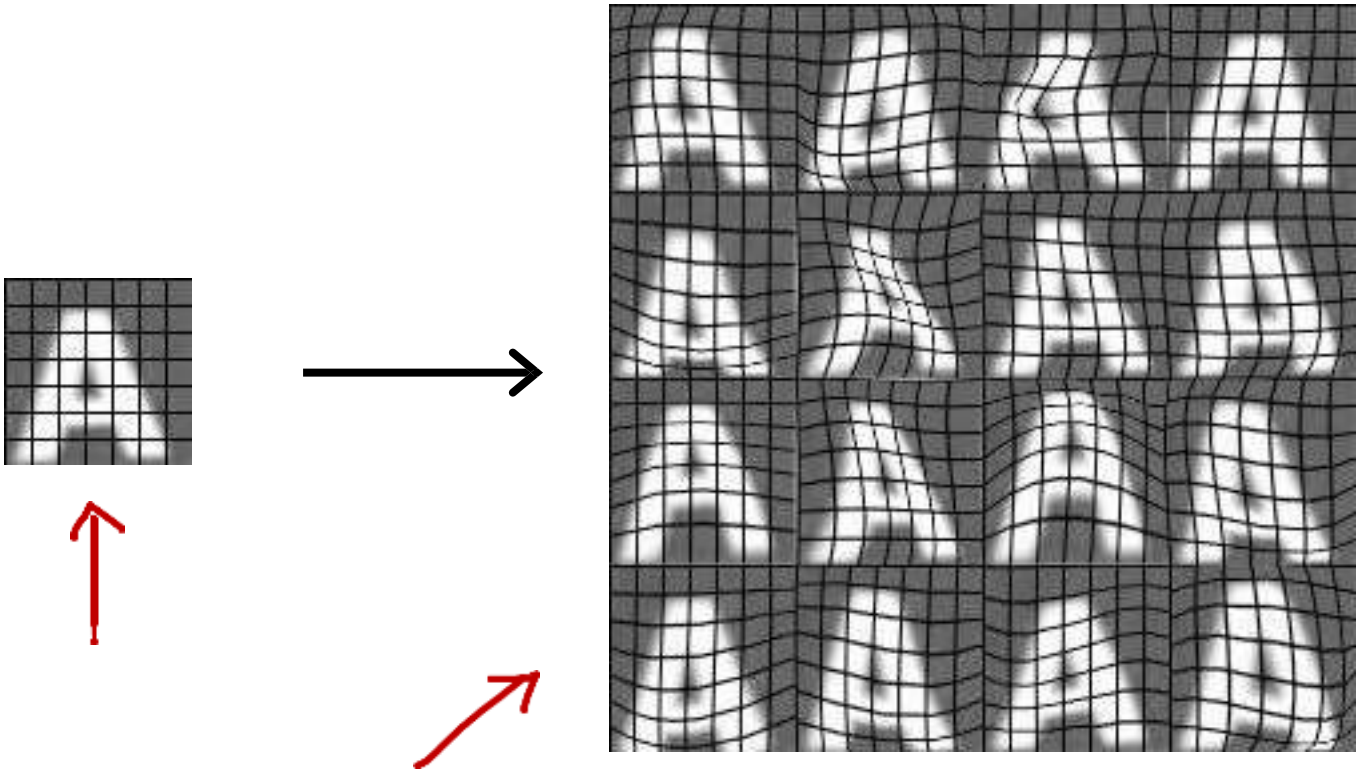
(x, y)

Data augmentation

Augmentation: modifying an existing training example to create a new training example.



Data augmentation by introducing distortions



Data augmentation for speech

Speech recognition example


Original audio (voice search: ”What is today’s weather?”)

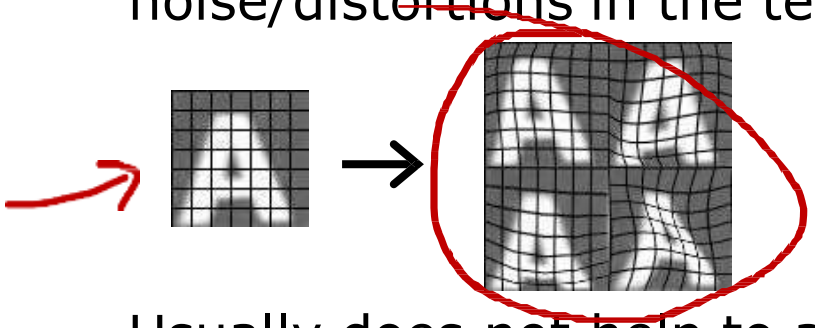

+ Noisy background: Crowd


+ Noisy background: Car


+ Audio on bad cellphone connection

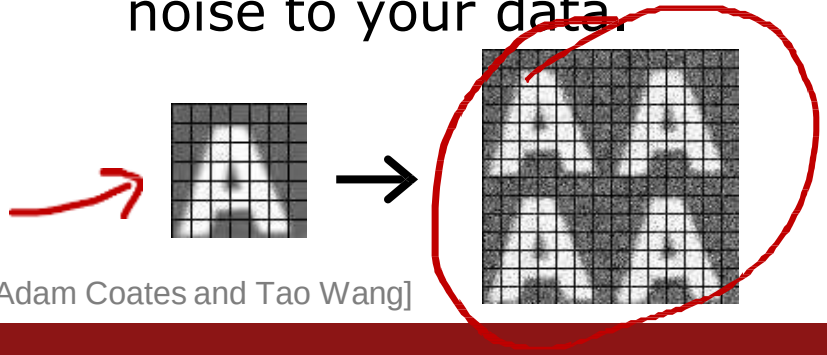
Data augmentation by introducing distortions

Distortion introduced should be representation of the type of noise/distortions in the test set.



Audio:
Background noise,
bad cellphone connection

Usually does not help to add purely random/meaningless noise to your data.



x_i=intensity (brightness) of pixel i
$x_i \leftarrow x_i +$ random noise

[Adam Coates and Tao Wang]

Data synthesis

Synthesis: using artificial data inputs to create a new training example.

Artificial data synthesis for photo OCR

[http://www.publicdomainpictures.net/view-image.php?image=5745&picture=times-square]

Artificial data synthesis for photo OCR

Real data

Abcdefg

Abcdefg

Abcdefg

Abcdefg

Abcdefg

[Adam Coates and Tao Wang]

Artificial data synthesis for photo OCR

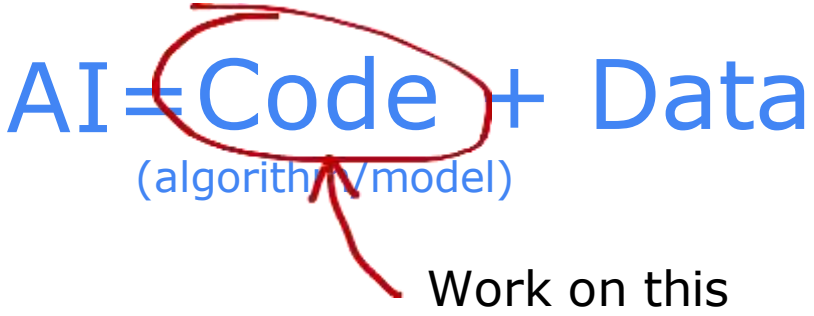
Real data

Synthetic data

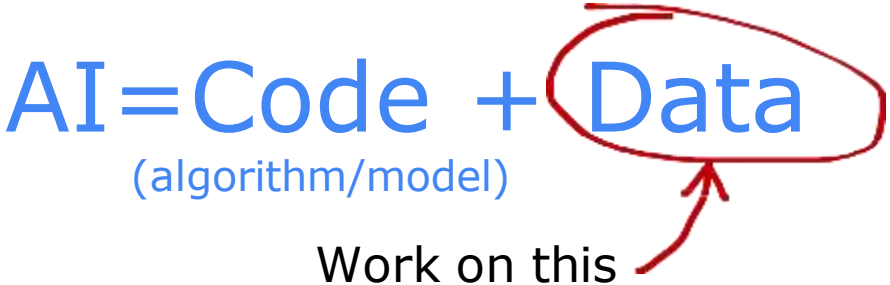
[Adam Coates and Tao Wang]

Engineering the data used by your system

Conventional model-centric approach:



Data-centric approach:



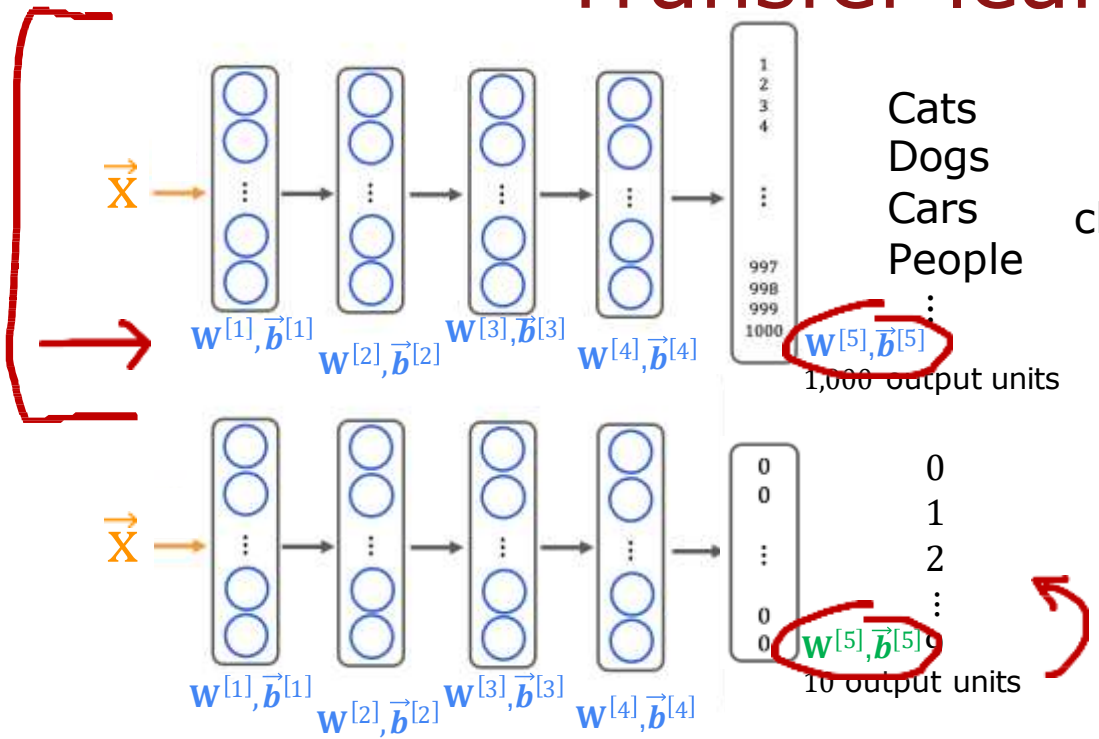
DeepLearning.AI

Stanford
ONLINE

Machine learning development process

Transfer learning: using data from a different task

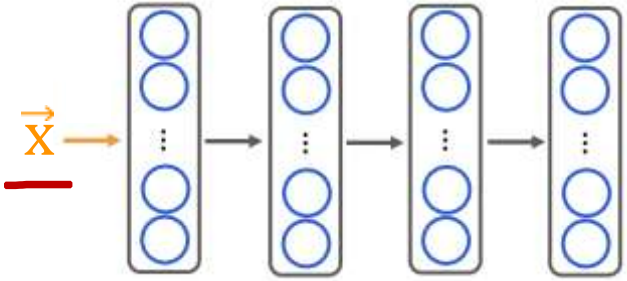
Transfer learning



Option 1: only train output layers parameters.

Option 2: train all parameters.

Why does transfer learning work?

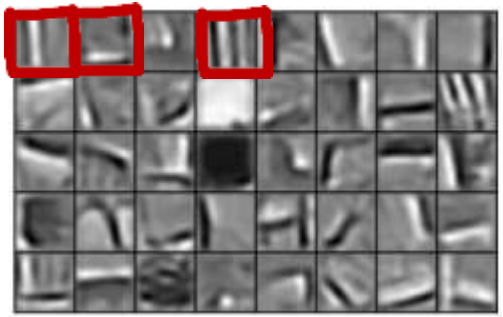


detects edges

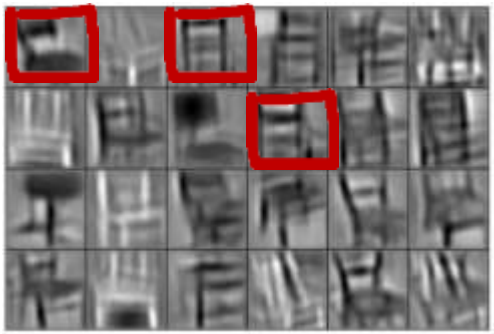
detects corners

detects curves/basic shapes

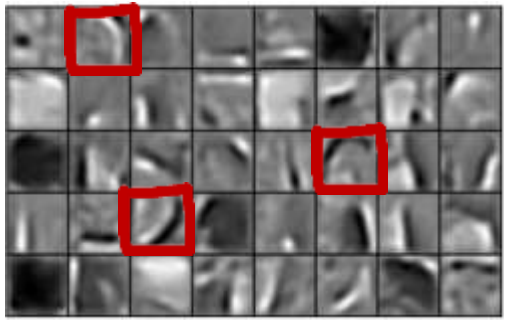
use the same input type



Edges



Corners



Curves / basic shapes

Transfer learning summary

1. Download neural network parameters pretrained on a large dataset with same input type (e.g., images, audio, text) as your application (or train your own).

1M

2. Further train (fine tune) the network on your own data.

1000

50

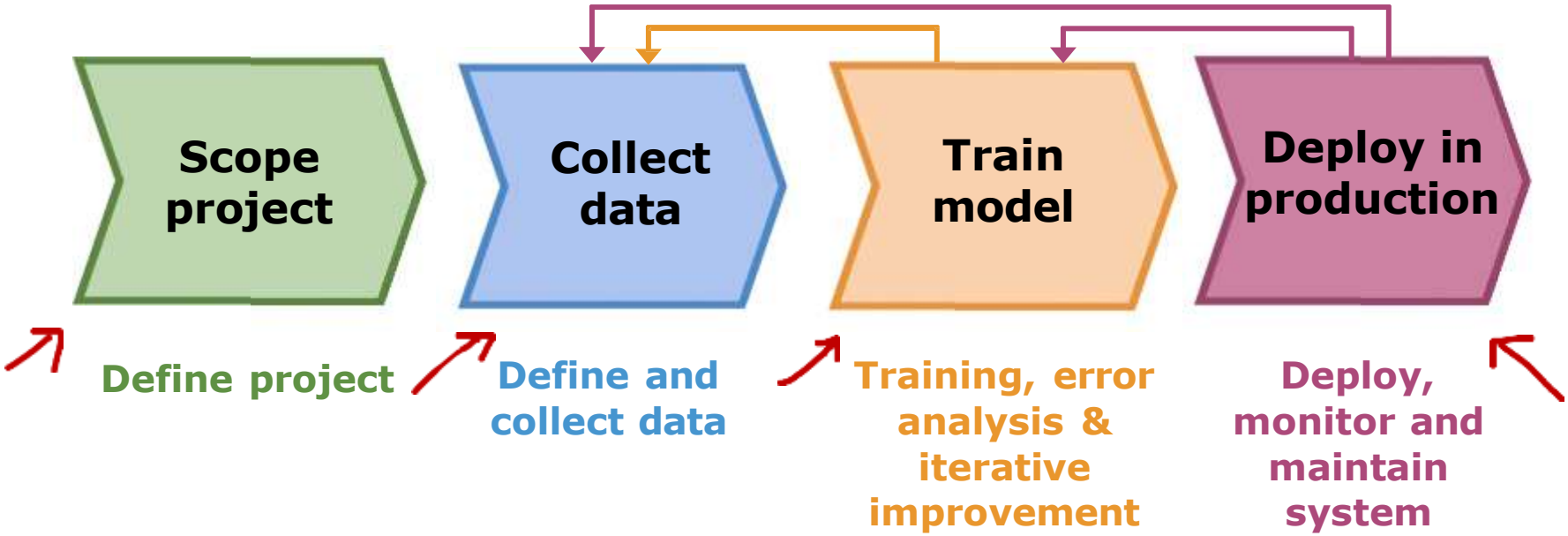
DeepLearning.AI

Stanford
ONLINE

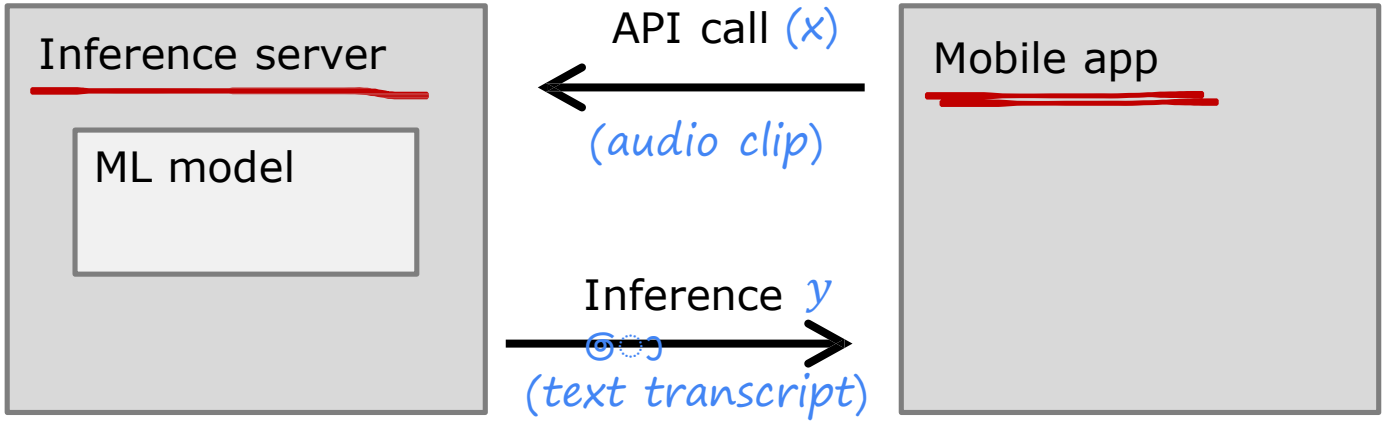
Machine learning development process

Full cycle of a machine learning project

Full cycle of a machine learning project



Deployment



- Software engineering may be needed for:
  - Ensure reliable and efficient predictions
  - Scaling
  - Logging
  - System monitoring
  - Model updates

MLOps
machine learning operations

DeepLearning.AI

Stanford
ONLINE

Machine learning development process

Fairness, bias, and ethics

Bias

Hiring tool that discriminates against women.

Facial recognition system matching dark skinned individuals to criminal mugshots.

Biased bank loan approvals.

Toxic effect of reinforcing negative stereotypes.

Adverse use cases

Deepfakes

Spreading toxic/incendiary speech through optimizing for engagement.

Generating fake content for commercial or political purposes.

Using ML to build harmful products, commit fraud etc.

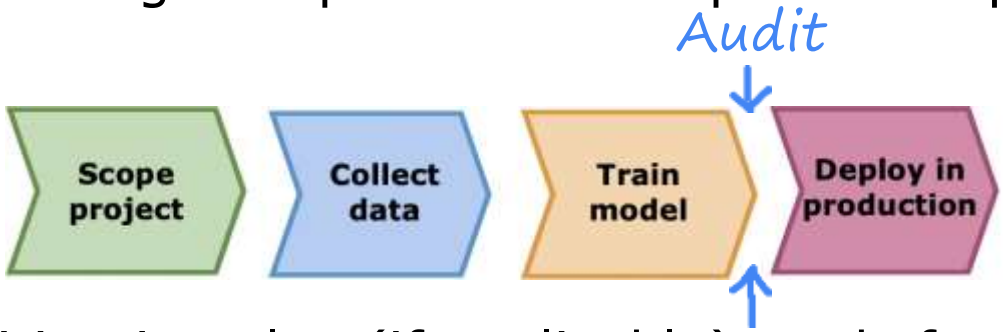
Spam vs anti-spam : fraud vs anti-fraud.

Guidelines

Get a diverse team to brainstorm things that might go wrong, with emphasis on possible harm to vulnerable groups.

Carry out literature search on standards/guidelines for your industry.

Audit systems against possible harm prior to deployment.



Develop mitigation plan (if applicable), and after deployment, monitor for possible harm.

DeepLearning.AI

Stanford
ONLINE

Skewed datasets (optional)

Error metrics for skewed datasets

Rare disease classification example

Train classifier $f_{\vec{w},b}(\vec{x})$

($y = 1$ if disease present,
$y = 0$ otherwise)

Find that you've got 1% error on test set
(99% correct diagnoses)

Only 0.5% of patients have the disease

```
print("y=0")
```

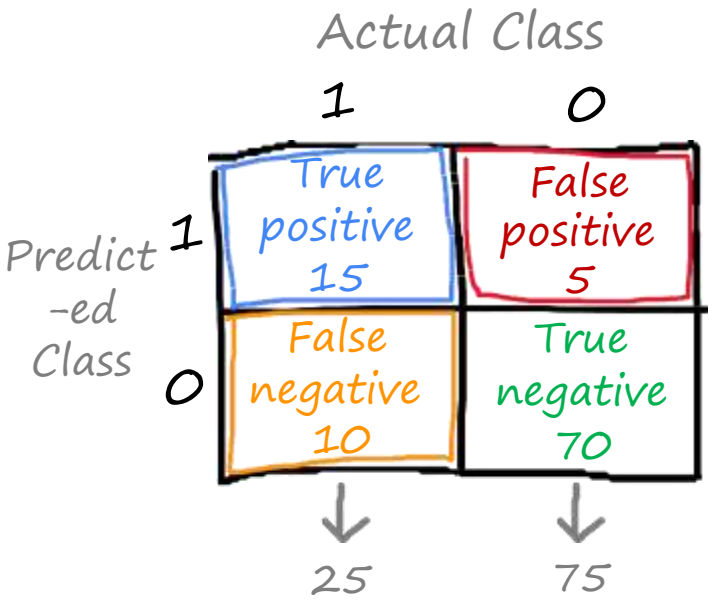
99.5% accuracy, 0.5% error ←

1%

1.2% ←

Precision/recall

$y = 1$ in presence of rare class we want to detect.



Precision:
(of all patients where we predicted $y = 1$, what fraction actually have the rare disease?)

$$\frac{\text{True positives}}{\#\text{predicted positive}} = \frac{\text{True positives}}{\text{True pos} + \text{False pos}} = \frac{15}{15+5} = 0.75$$

Recall:
(of all patients that actually have the rare disease, what fraction did we correctly detect as having it?)

$$\frac{\text{True positives}}{\#\text{actual positive}} = \frac{\text{True positives}}{\text{True pos} + \text{False neg}} = \frac{15}{15+10} = 0.6$$

```
print("y=0")
```

DeepLearning.AI

Stanford
ONLINE

Skewed datasets (optional)

Trading off precision and recall

Trading off precision and recall

Logistic regression: $0 < f_{\vec{w},b}(\vec{x}) < 1$

Predict 1 if $f_{\vec{w},b}(\vec{x}) \geq 0.5$ ~~0.7~~ ~~0.9~~ 0.3

Predict 0 if $f_{\vec{w},b}(\vec{x}) < 0.5$ ~~0.7~~ ~~0.9~~ 0.3

$$\text{precision} = \frac{\text{true positives}}{\text{total predicted positive}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{total actual positive}}$$

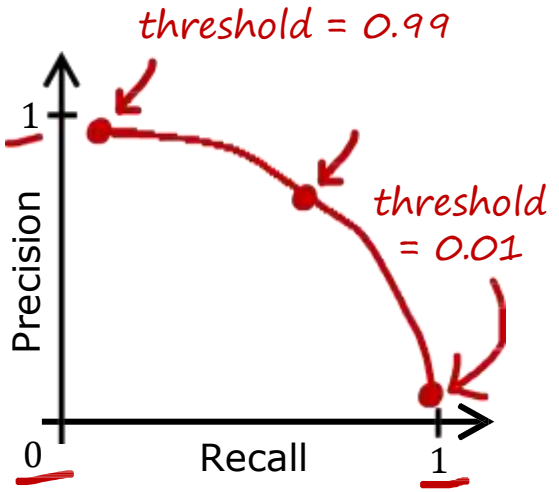
Suppose we want to predict $y = 1$ (rare disease) only if very confident.

→ higher precision, lower recall.

Suppose we want to avoid missing too many cases of rare disease (when in doubt predict $y = 1$)

→ lower precision, higher recall.

More generally predict 1 if: $f_{\vec{w},b}(\vec{x}) \geq$ threshold.



F1 score

How to compare precision/recall numbers?

	Precision (P)	Recall (R)	Average	F_1 score
Algorithm 1	0.5	0.4	0.45	0.444
Algorithm 2	0.7	0.1	0.4	0.175
Algorithm 3	0.02	1.0	0.501	0.0392

`print("y=1")`

Average $= \frac{P+R}{2}$

F1 score $= \frac{1}{\frac{1}{2}\left(\frac{1}{P}+\frac{1}{R}\right)} = 2\frac{PR}{P+R}$

Copyright Notice

DeepLearning.AI

Stanford
ONLINE

Decision Trees

Decision Tree Model

Cat classification example

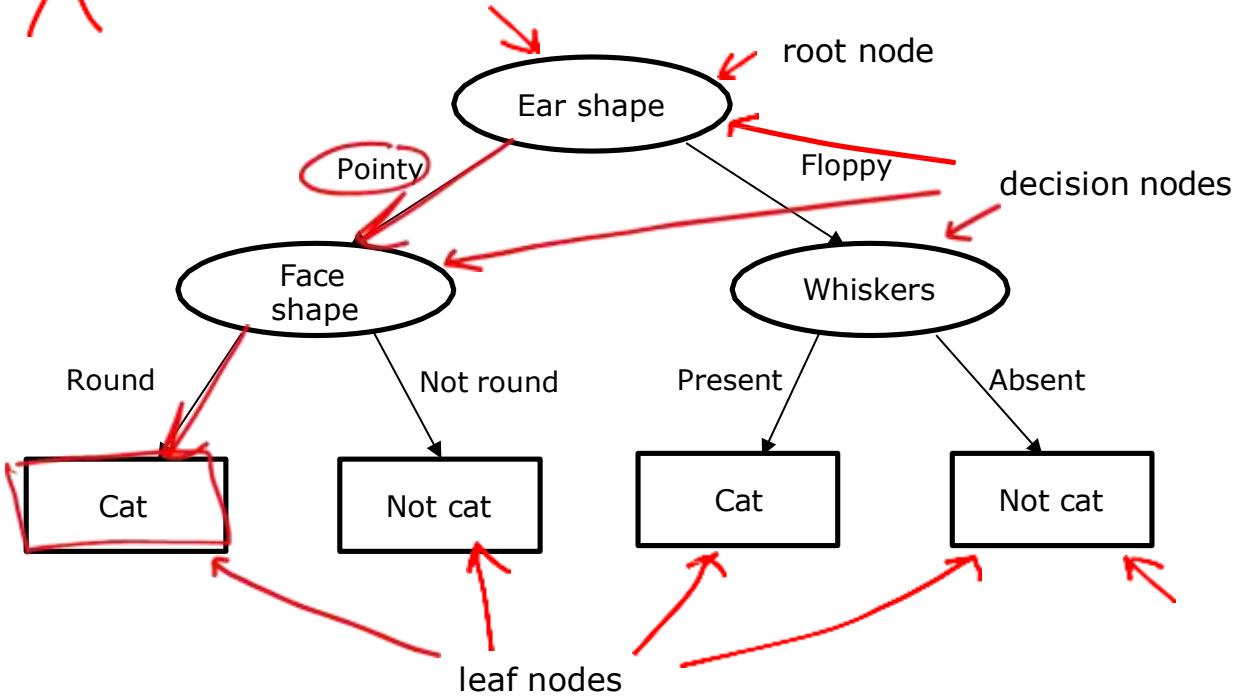
	Ear shape (x_1)	Face shape(x_2)	Whiskers (x_3)	Cat
	Pointy	Round	Present	1
	Floppy	Not round	Present	1
	Floppy	Round	Absent	0
	Pointy	Not round	Present	0
	Pointy	Round	Present	1
	Pointy	Round	Absent	1
	Floppy	Not round	Absent	0
	Pointy	Round	Absent	1
	Floppy	Round	Absent	0
	Floppy	Round	Absent	0

Categorical (discrete values)

X

y

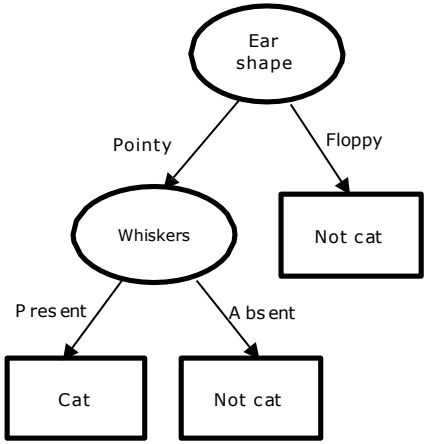
Decision Tree

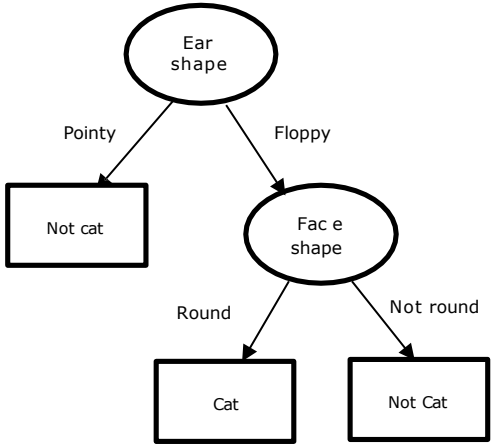


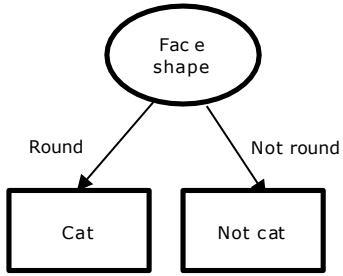
New test example

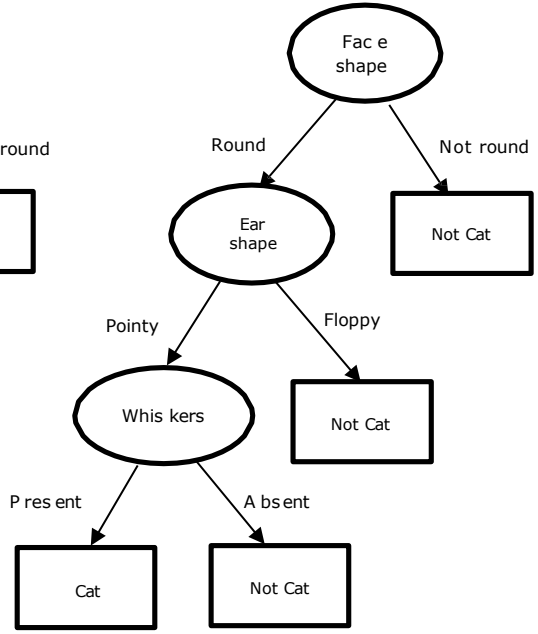
Ear shape: Pointy
Face shape: Round
Whiskers: Present

Decision Tree









DeepLearning.AI

Stanford
ONLINE

Decision Trees

Learning Process

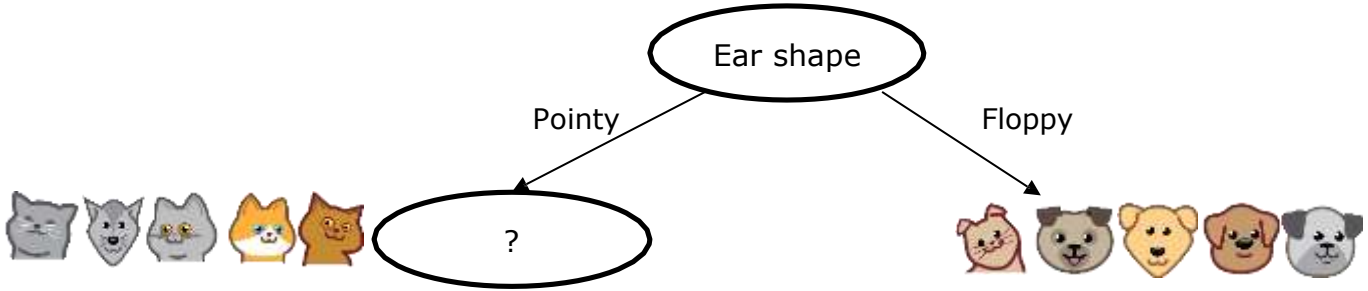
Decision Tree Learning

?

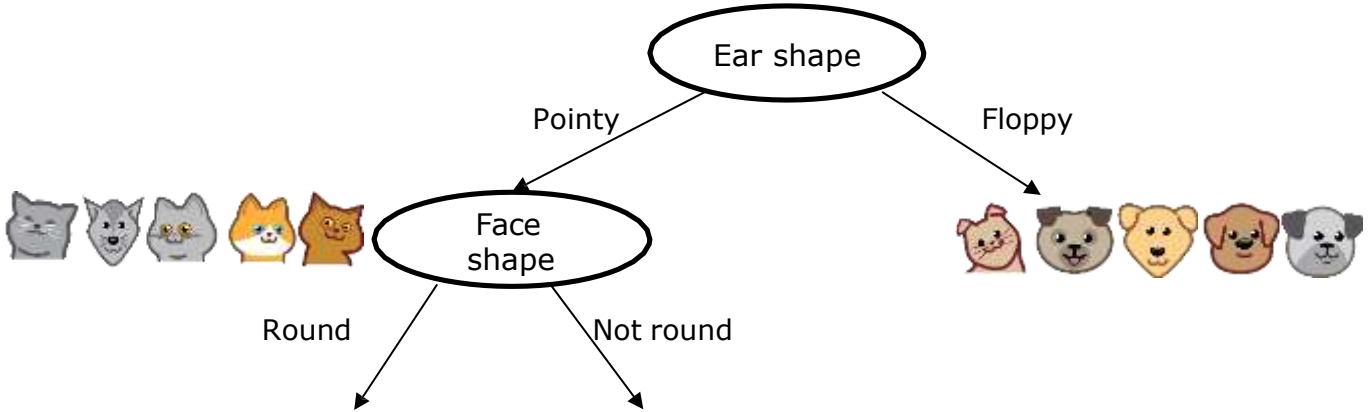
Decision Tree Learning

Decision Tree Learning

Decision Tree Learning

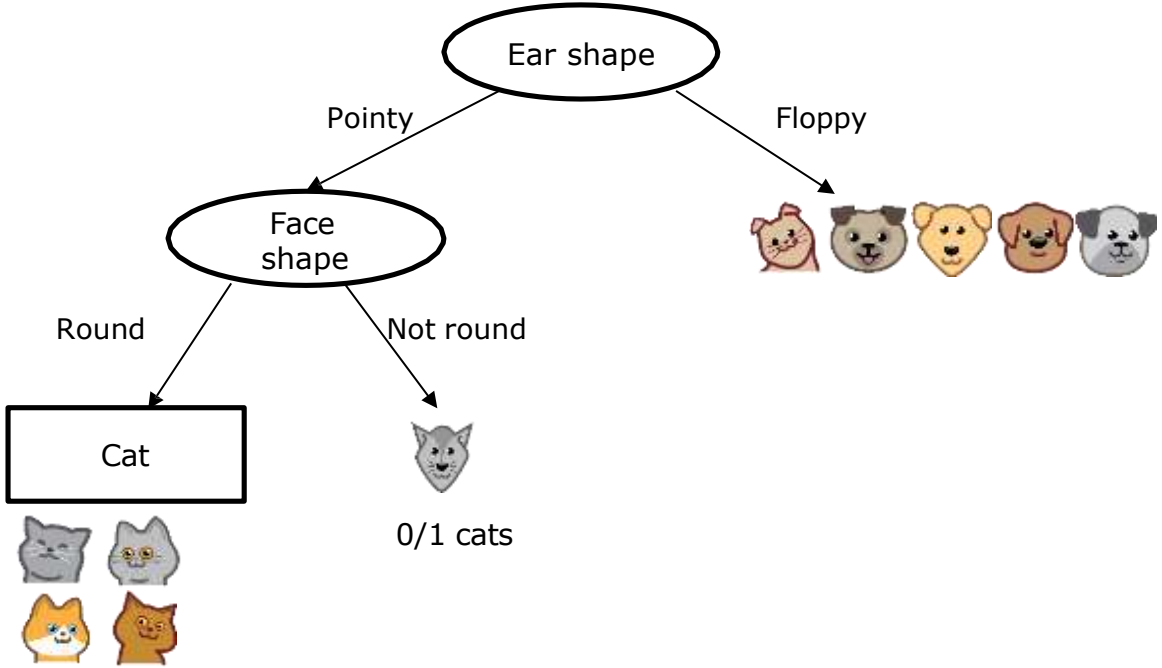


Decision Tree Learning

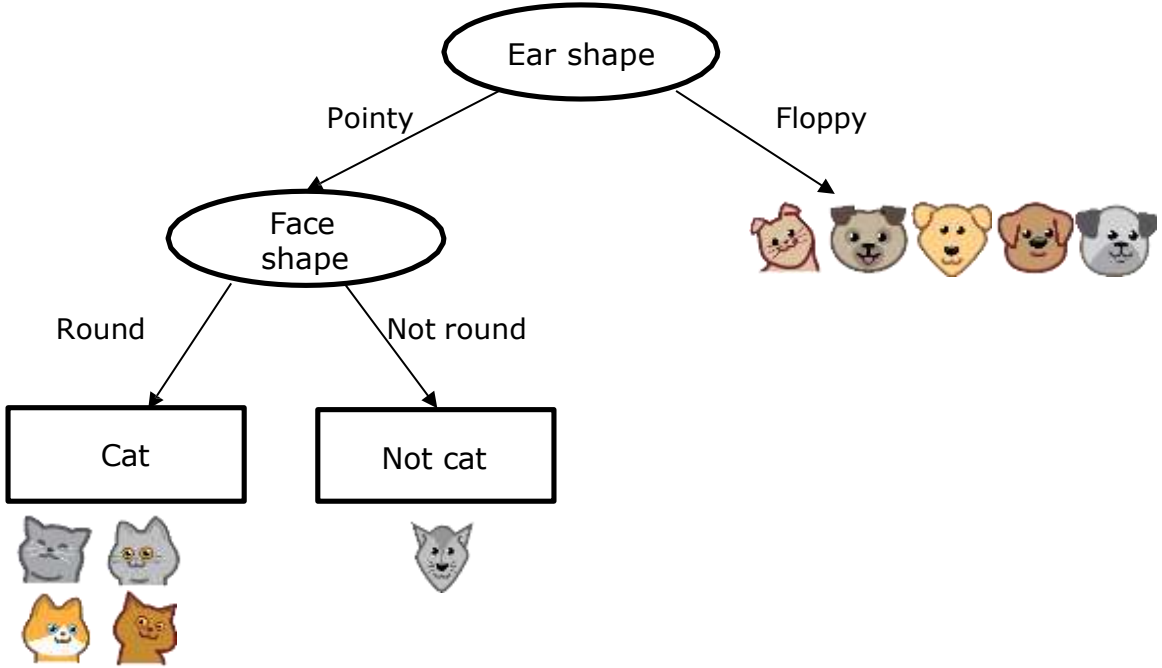


4/4 cats

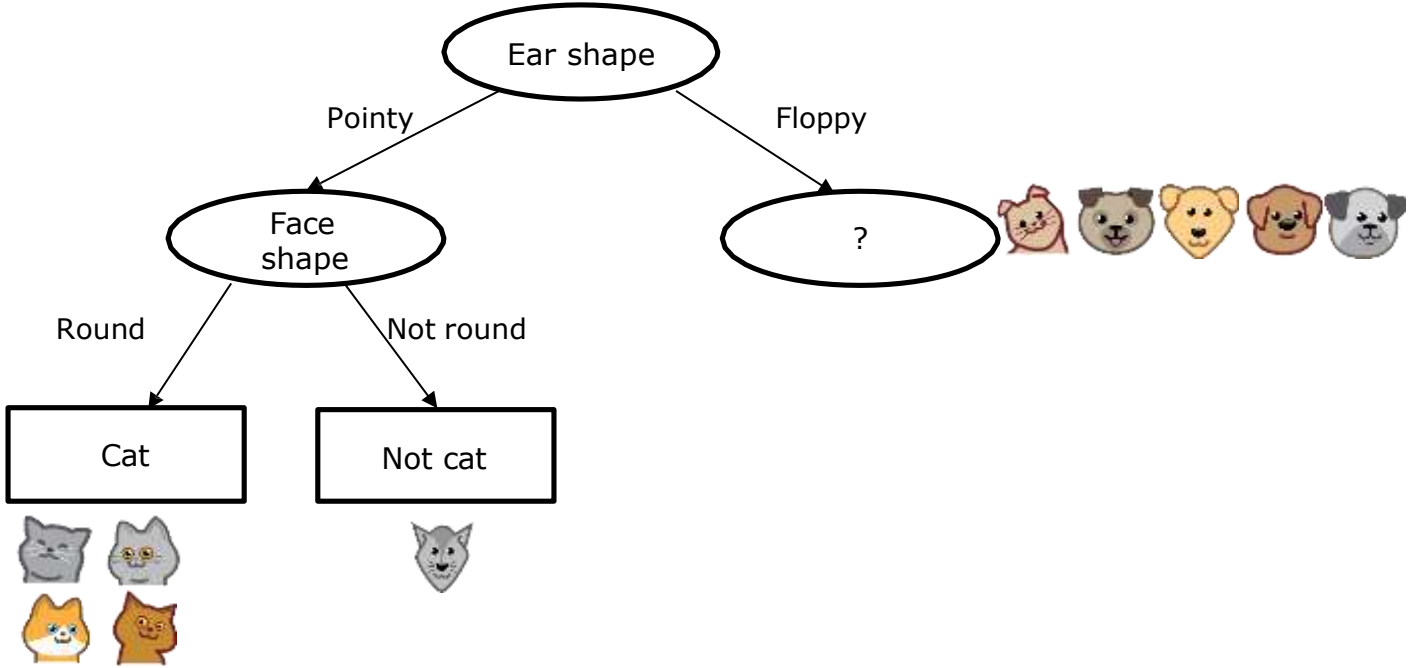
Decision Tree Learning



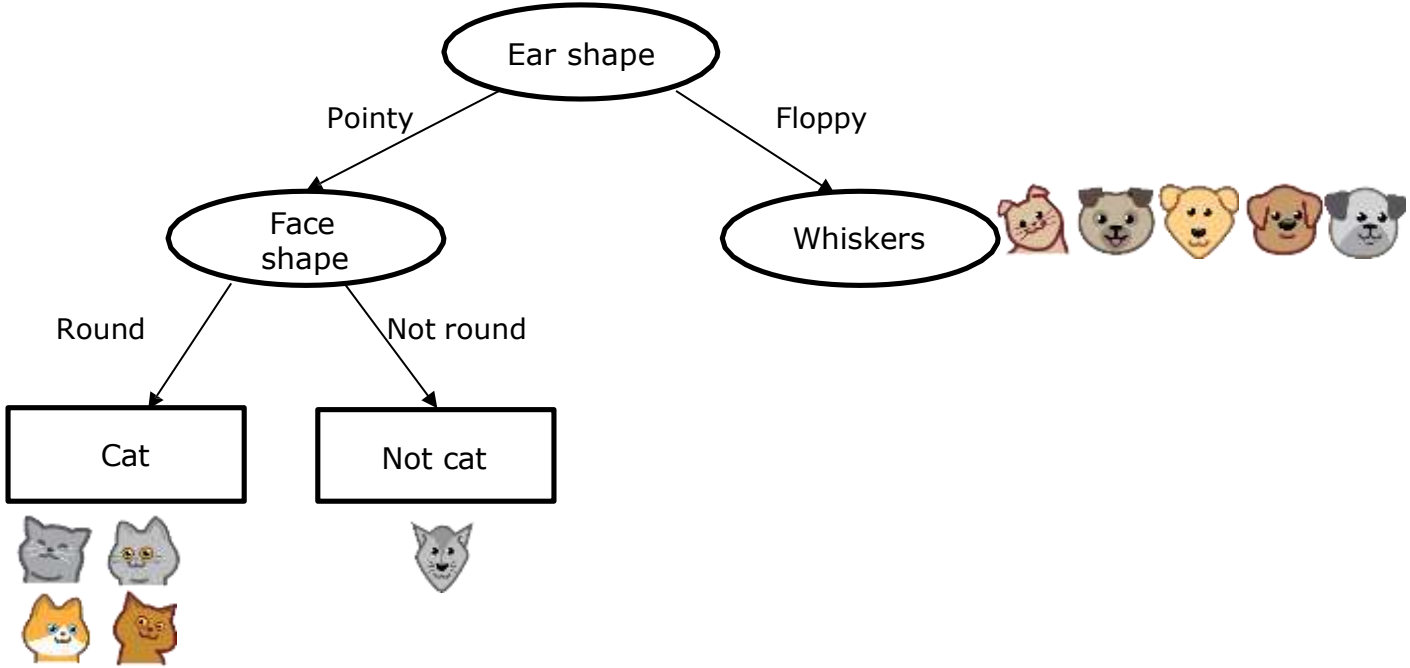
Decision Tree Learning



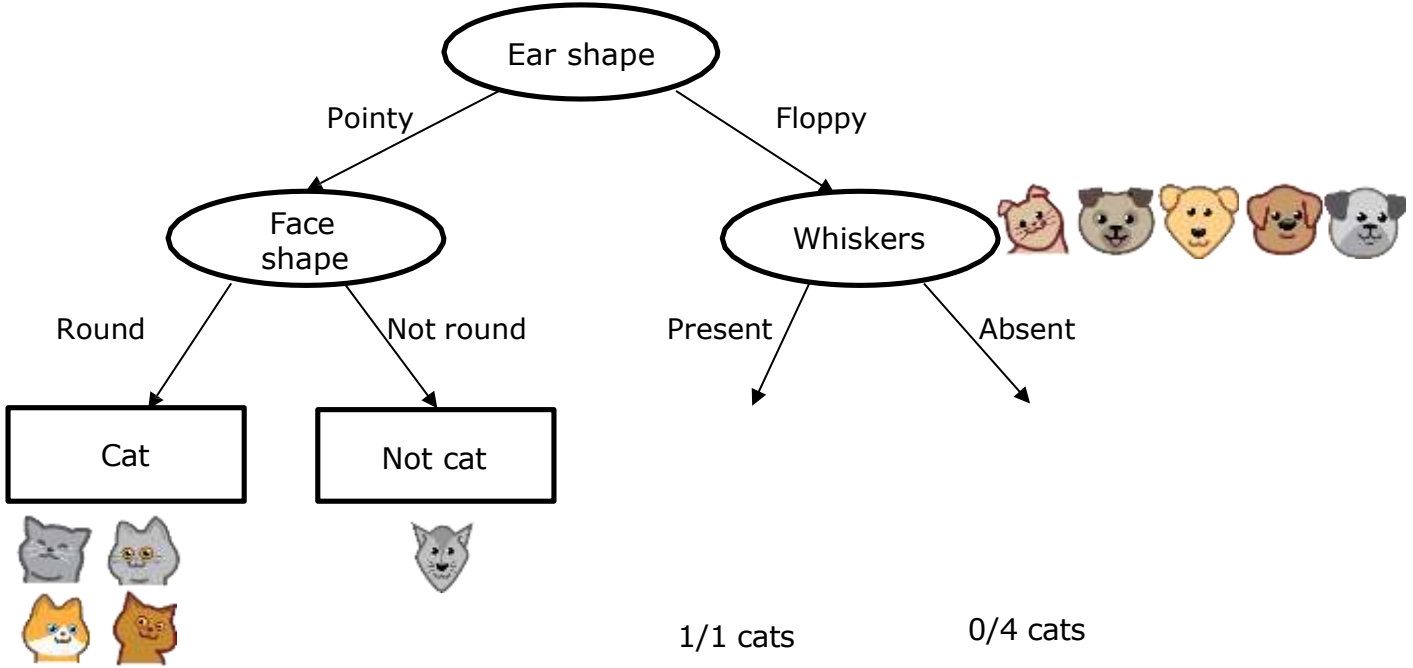
Decision Tree Learning



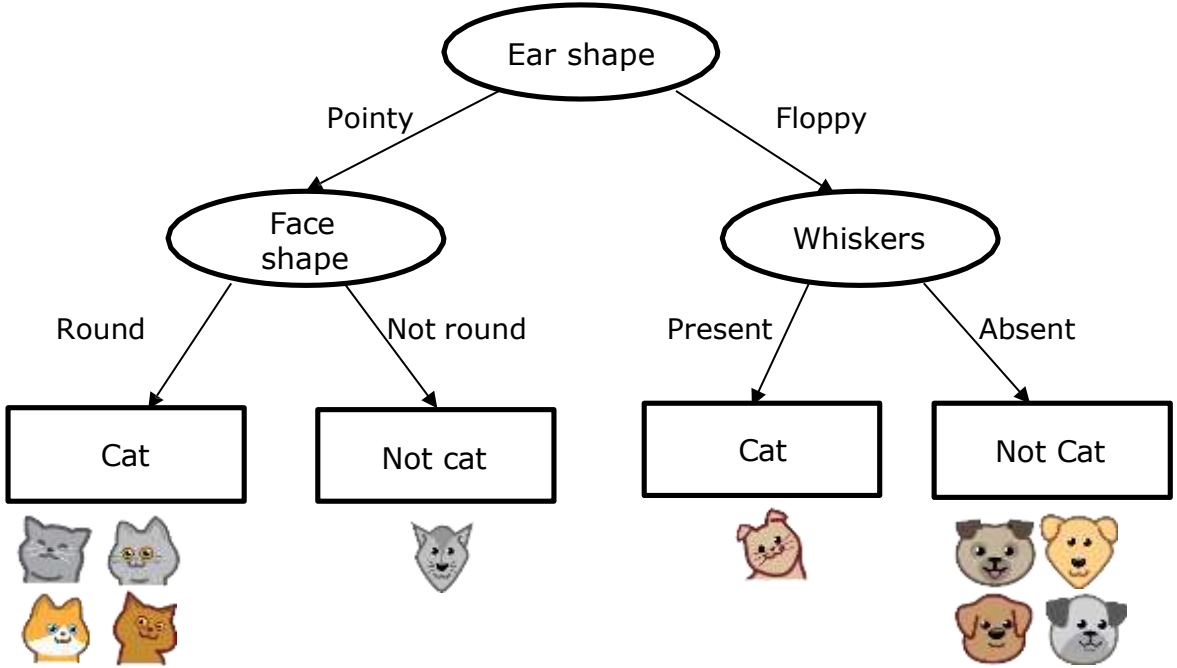
Decision Tree Learning



Decision Tree Learning



Decision Tree Learning



Decision Tree Learning

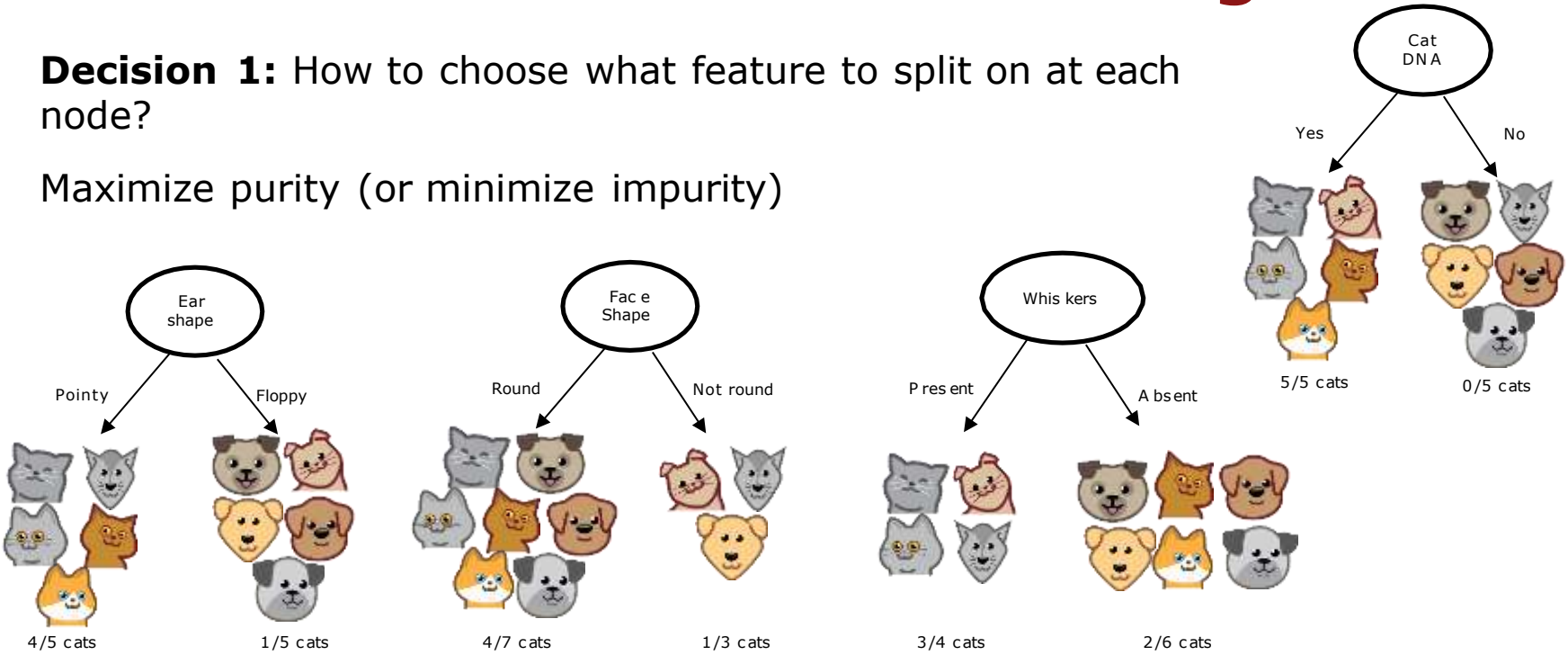
Decision 1: How to choose what feature to split on at each node?

Maximize purity (or minimize impurity)

Decision Tree Learning

Decision 1: How to choose what feature to split on at each node?

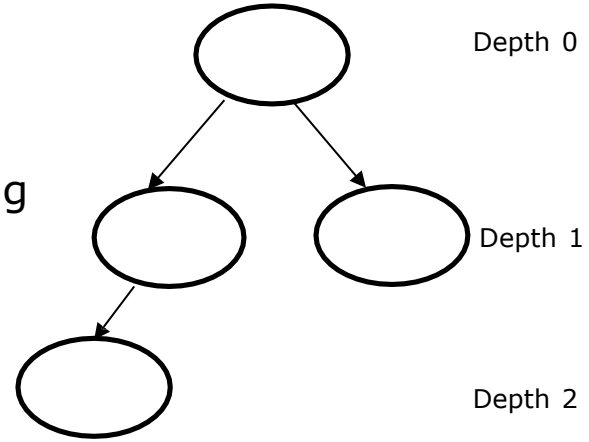
Maximize purity (or minimize impurity)



Decision Tree Learning

Decision 2: When do you stop splitting?

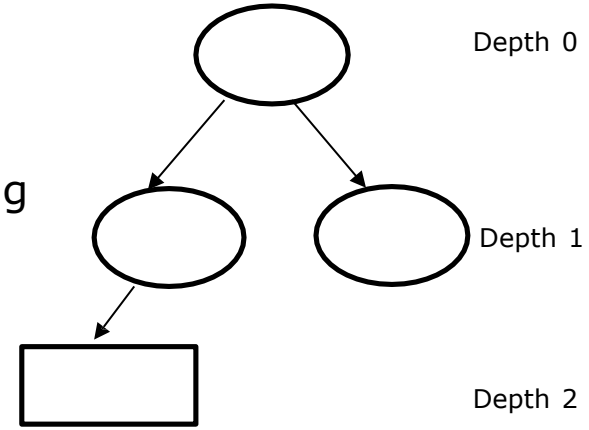
- When a node is 100% one class
- When splitting a node will result in the tree exceeding a maximum depth



Decision Tree Learning

Decision 2: When do you stop splitting?

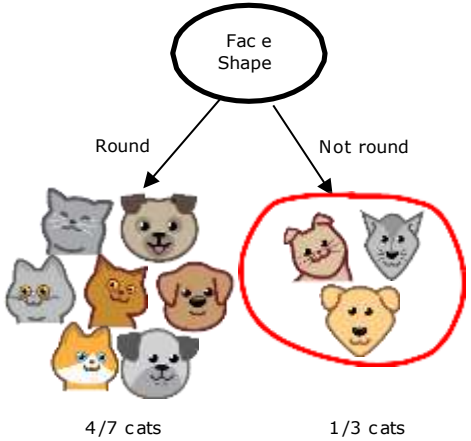
- When a node is 100% one class
- When splitting a node will result in the tree exceeding a maximum depth



Decision Tree Learning

Decision 2: When do you stop splitting?

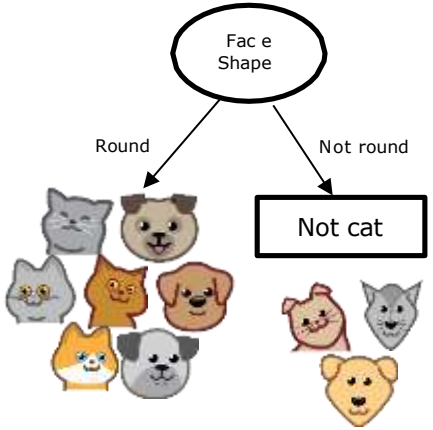
- When a node is 100% one class
- When splitting a node will result in the tree exceeding a maximum depth
- When improvements in purity score are below a threshold
- When number of examples in a node is below a threshold



Decision Tree Learning

Decision 2: When do you stop splitting?

- When a node is 100% one class
- When splitting a node will result in the tree exceeding a maximum depth
- When improvements in purity score are below a threshold
- When number of examples in a node is below a threshold



DeepLearning.AI

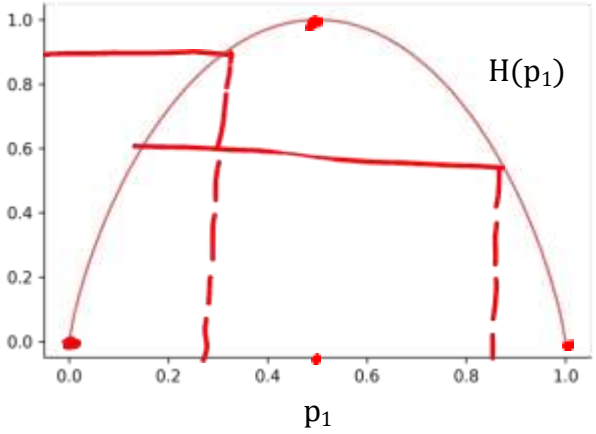
Stanford
ONLINE

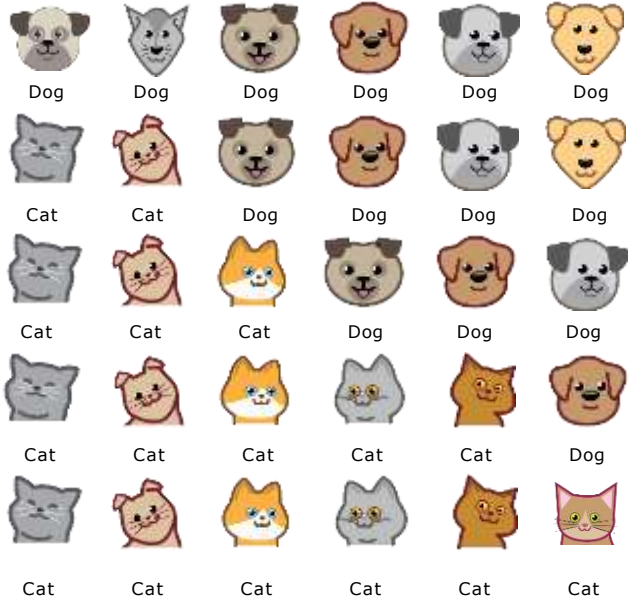
Decision Tree Learning

Measuring purity

Entropy as a measure of impurity

p_1 = fraction of examples that are cats





$p_1 = 0 \quad H(p_1) = 0$

$p_1 = 2/6 \quad H(p_1) = 0.92$

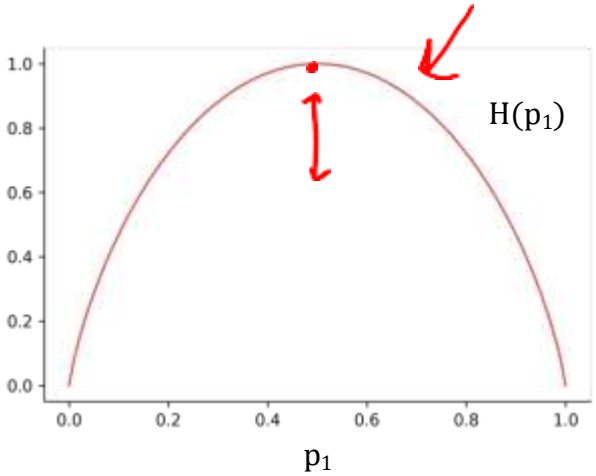
$p_1 = 3/6 \quad H(p_1) = 1$

$p_1 = 5/6 \quad H(p_1) = 0.65$

$p_1 = 6/6 \quad H(p_1) = 0$

Entropy as a measure of impurity

p_1 = fraction of examples that are cats



$$p_0 = 1 - p_1$$

$$H(p_1) = -p_1 log_2(p_1) - p_0 log_2(p_0)$$

$$= -p_1 \, log_2(p_1) - (1 - p_1) log_2(1 - p_1)$$

Note: "0 log(0)" = 0

DeepLearning.AI

Stanford
ONLINE

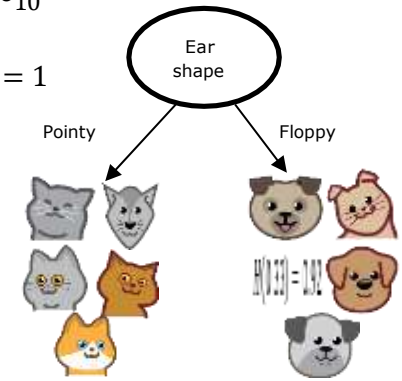
Decision Tree Learning

Choosing a split: Information Gain

Choosing a split

$p_1 = 5/10 = 0.5$

$H(0.5) = 1$



$p_1 = 4/5 = 0.8$

$H(0.8) = 0.72$

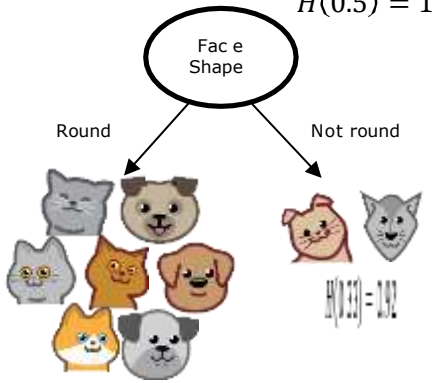
$p_1 = 1/5 = 0.2$

$H(0.2) = 0.72$

$H(0.5) - \left(\frac{5}{10}H(0.8) + \frac{5}{10}H(0.2)\right)$

$= 0.28$

$H(0.5) = 1$



$p_1 = 4/7 = 0.57$

$H(0.57) = 0.99$

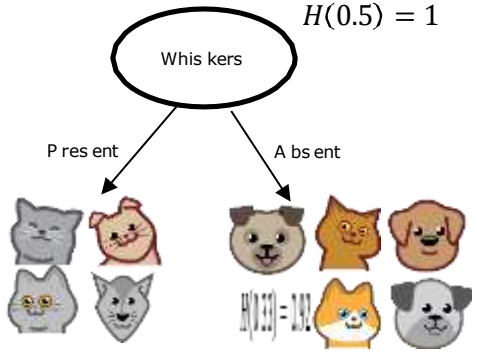
$p_1 = 1/3 = 0.33$

$H(0.33) = 0.92$

$H(0.5) - \left(\frac{7}{10}H(0.57) + \frac{3}{10}H(0.33)\right)$

$= 0.03$

$H(0.5) = 1$



$p_1 = 3/4 = 0.75$

$H(0.75) = 0.81$

$p_1 = 2/6 = 0.33$

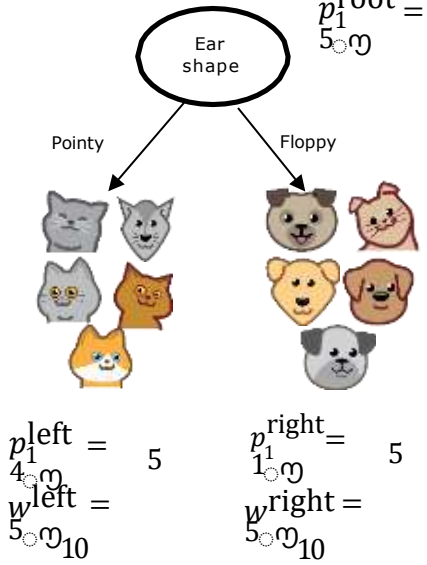
$H(0.33) = 0.92$

$H(0.5) - \left(\frac{4}{10}H(0.75) + \frac{6}{10}H(0.33)\right)$

$= 0.12$

Information Gain

$p_1^{\text{root}} = \frac{5}{10} = 0.5$



$p_1^{\text{left}} = \frac{4}{5}$ $w^{\text{left}} = \frac{5}{10}$

$p_1^{\text{right}} = \frac{1}{5}$ $w^{\text{right}} = \frac{5}{10}$

Information gain

$$= \mathrm{H}(p_1^{\text{root}}) - \left(w^{\text{left}} H\left(p_1^{\text{left}}\right) + w^{\text{right}} H\left(p_1^{\text{right}}\right) \right)$$

DeepLearning.AI

Stanford
ONLINE

Decision Tree Learning

Putting it together

Decision Tree Learning

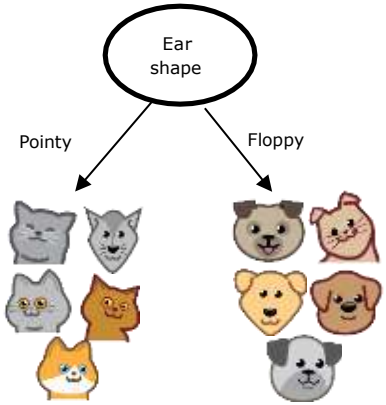
- Start with all examples at the root node
- Calculate information gain for all possible features, and pick the one with the highest information gain
- Split dataset according to selected feature, and create left and right branches of the tree
- Keep repeating splitting process until stopping criteria is met:
    - When a node is 100% one class
    - When splitting a node will result in the tree exceeding a maximum depth
    - Information gain from additional splits is less than threshold
    - When number of examples in a node is below a threshold

Recursive splitting

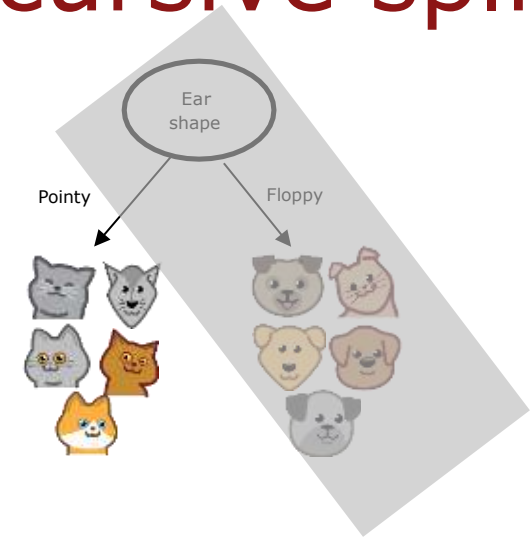
Recursive splitting

Recursive splitting

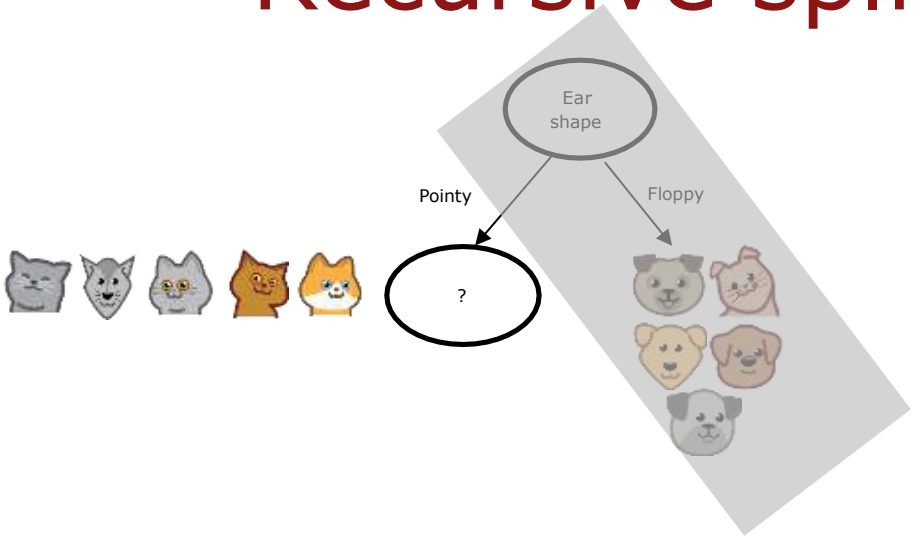
Recursive splitting



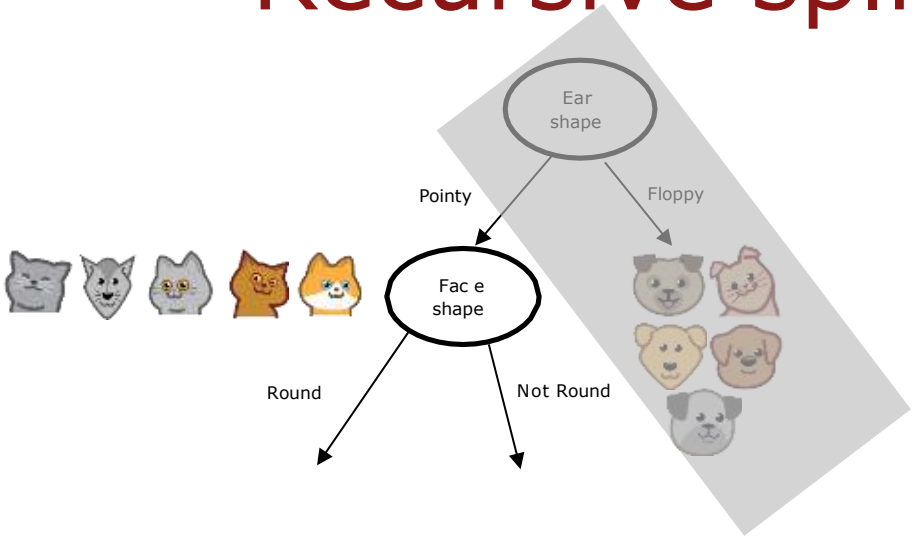
Recursive splitting



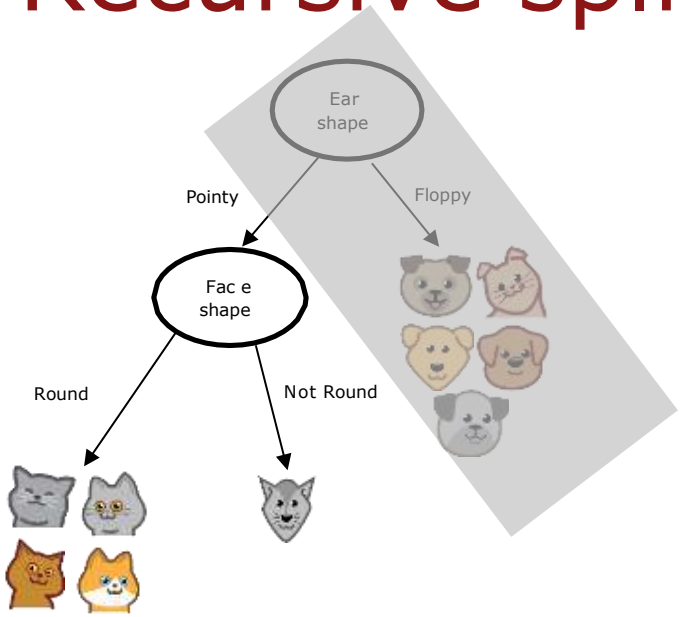
Recursive splitting



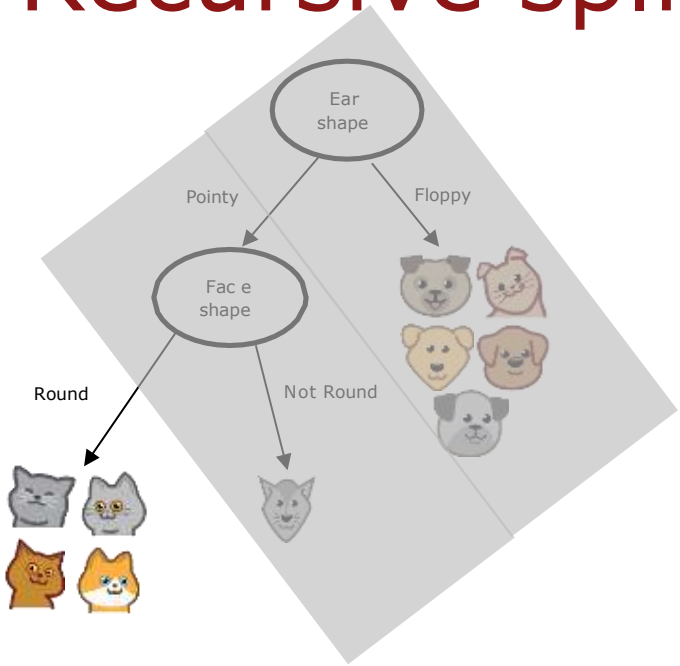
Recursive splitting



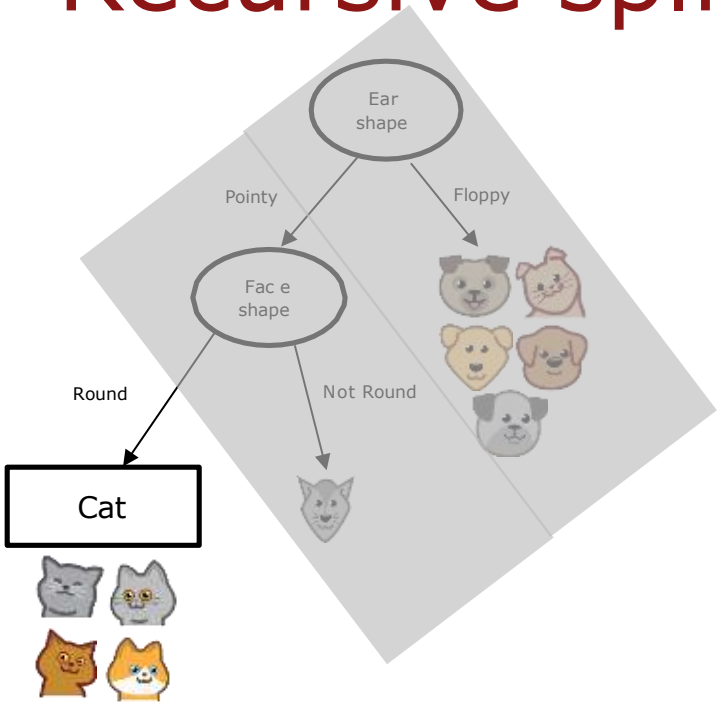
Recursive splitting



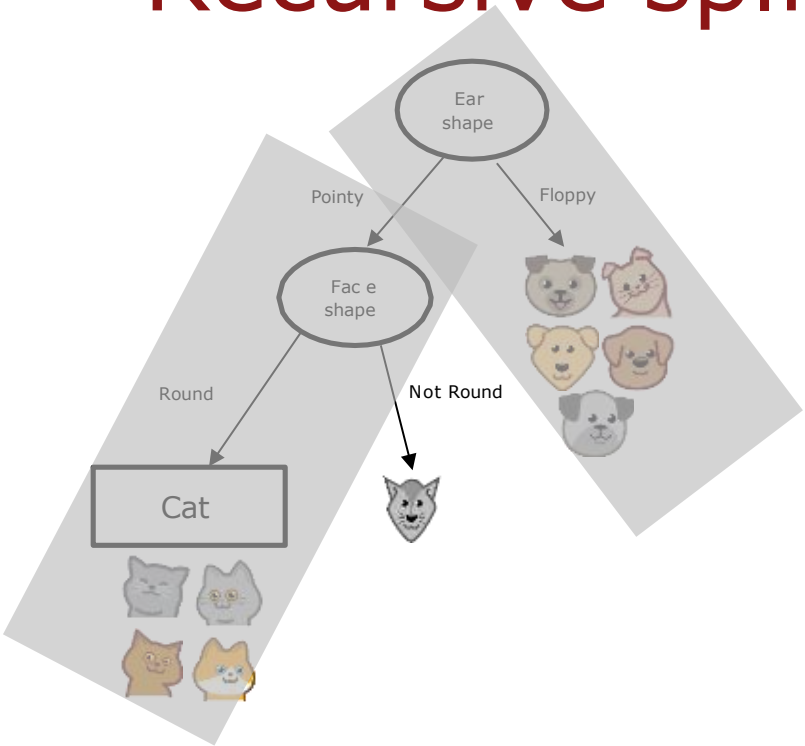
Recursive splitting



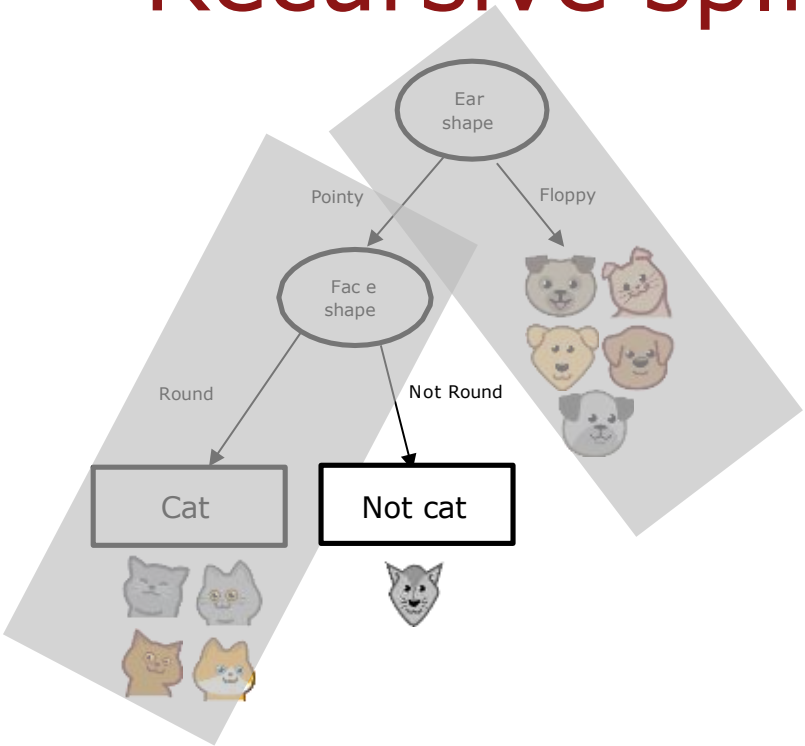
Recursive splitting



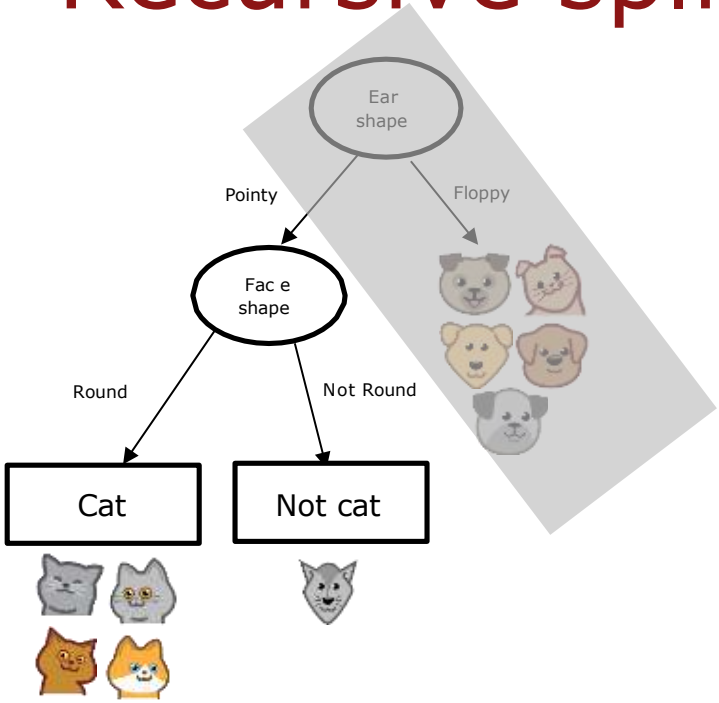
Recursive splitting



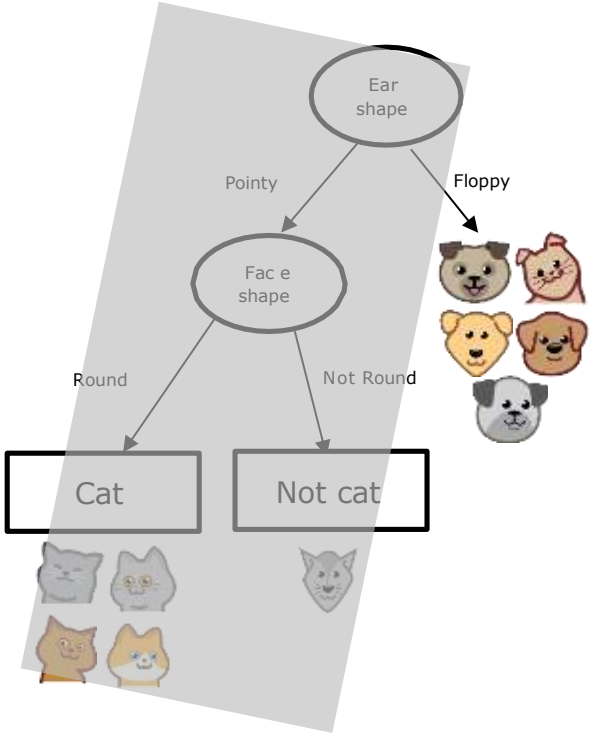
Recursive splitting



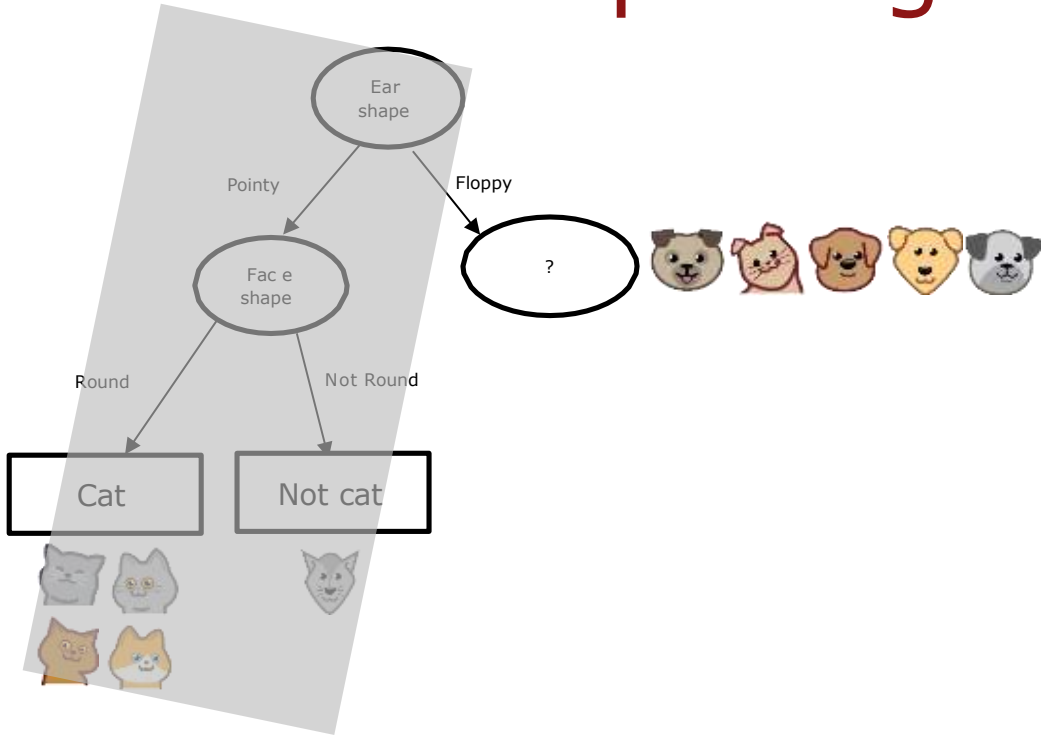
Recursive splitting



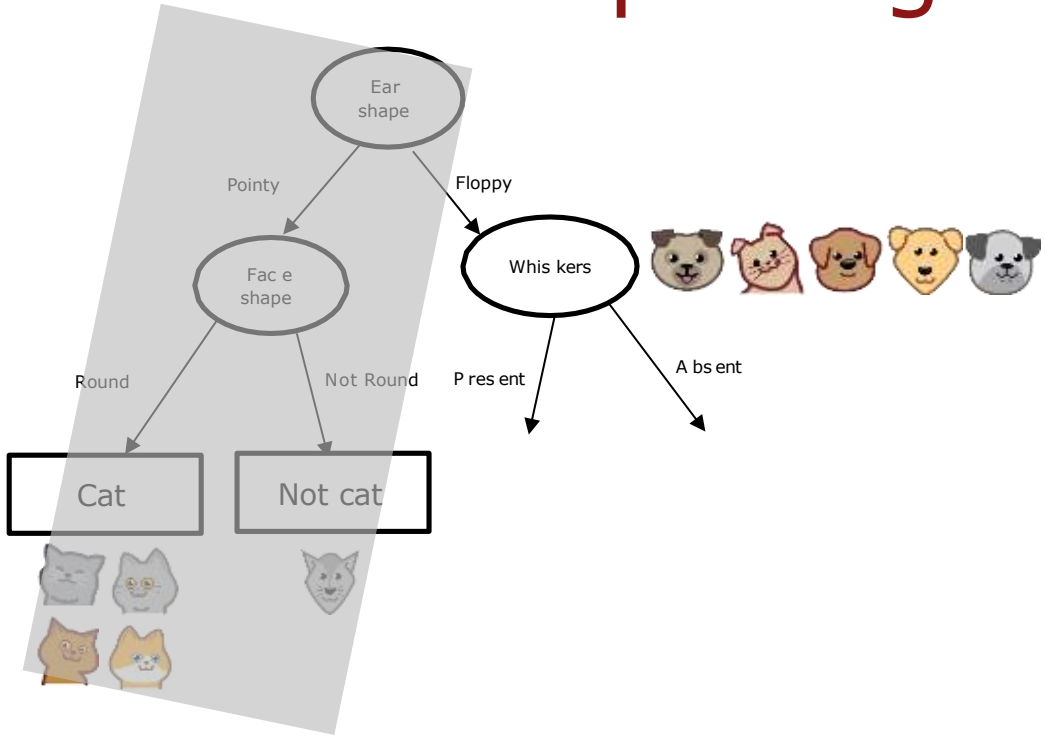
Recursive splitting



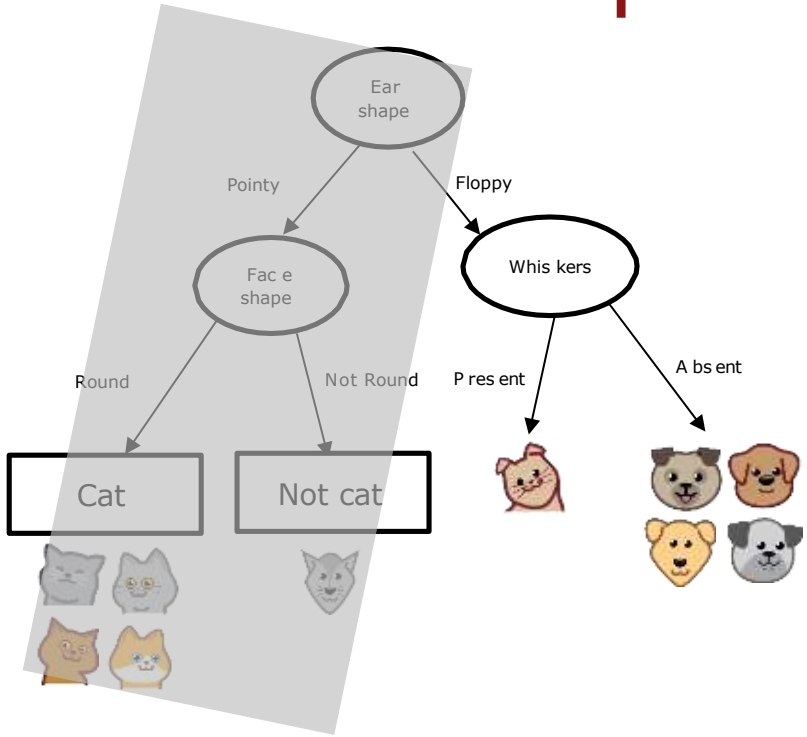
Recursive splitting



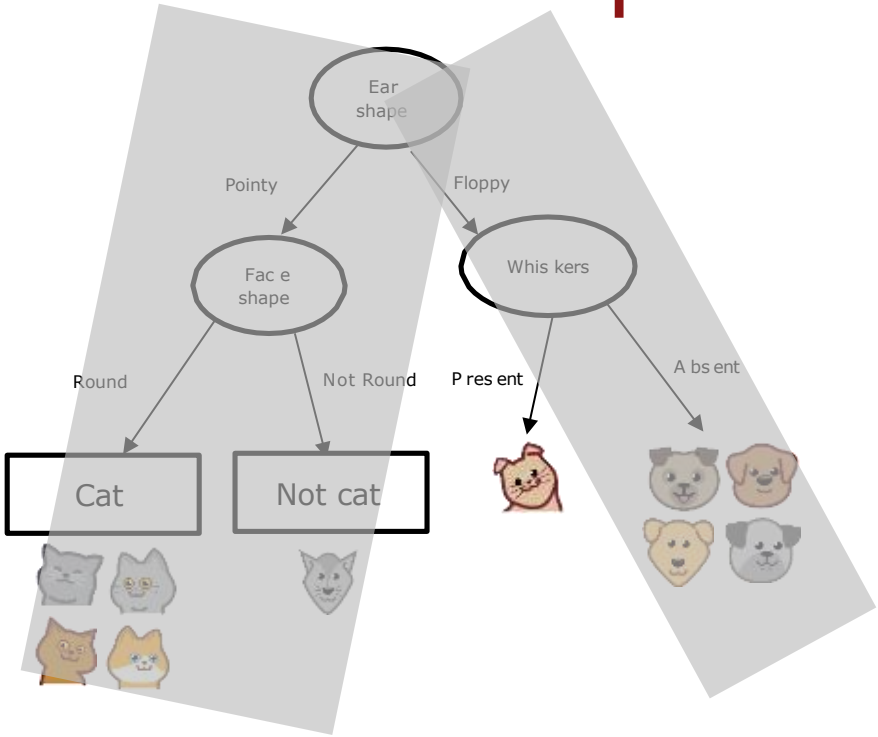
Recursive splitting



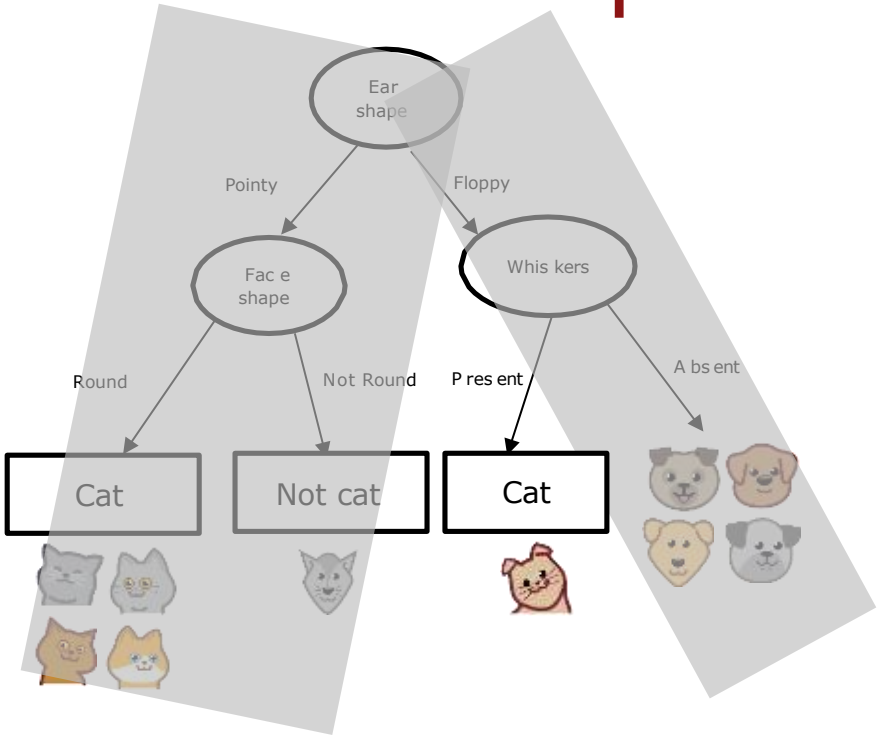
Recursive splitting



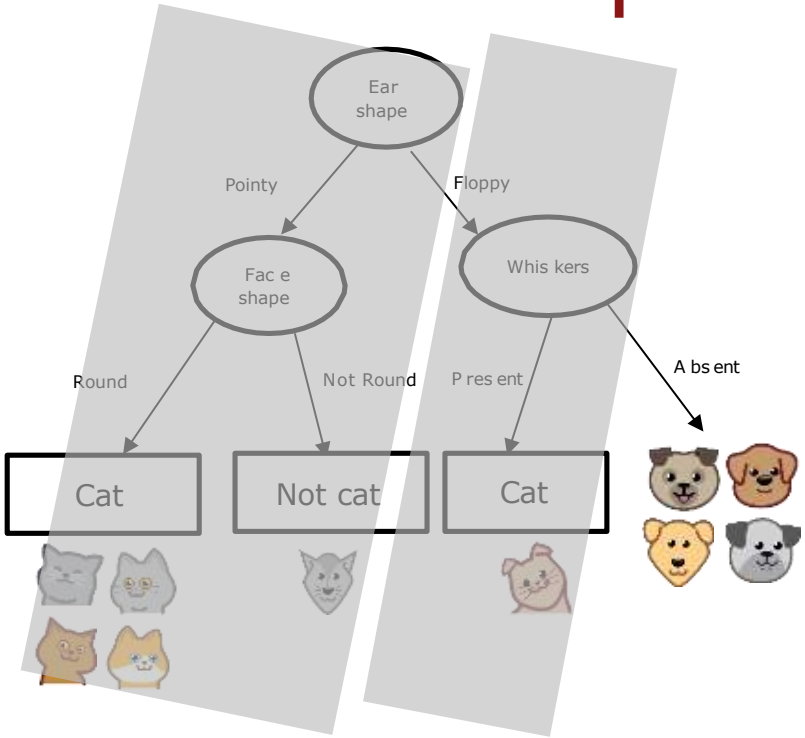
Recursive splitting



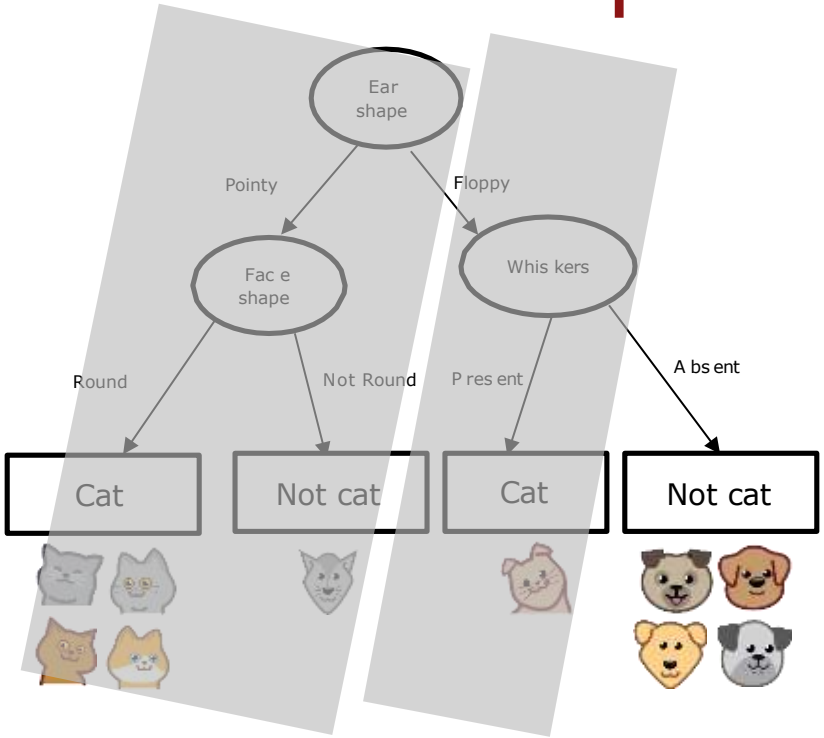
Recursive splitting



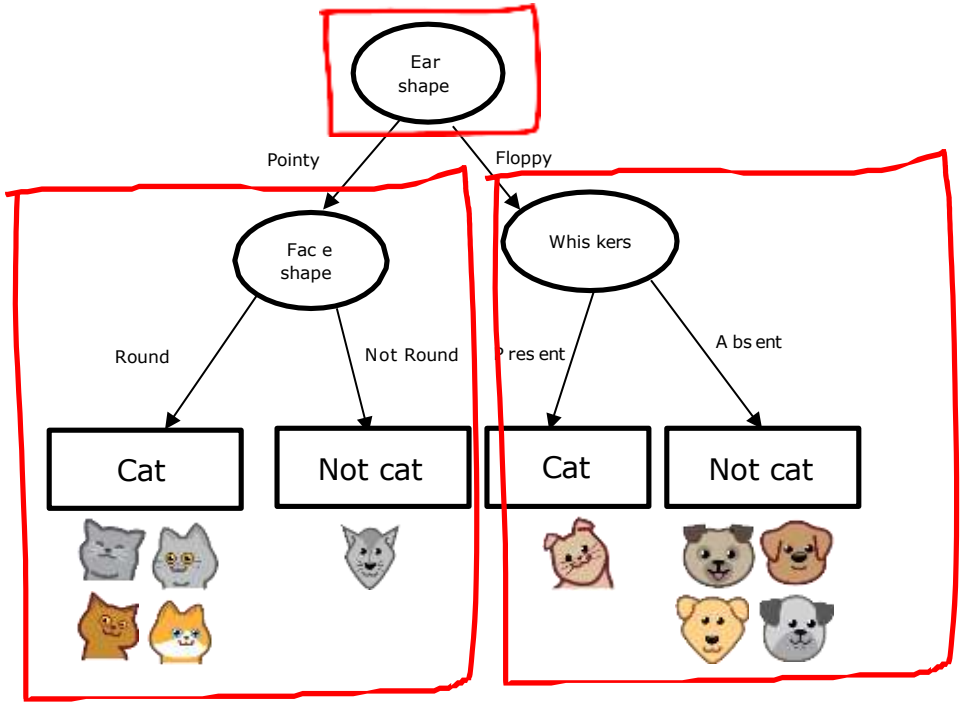
Recursive splitting



Recursive splitting



Recursive splitting



Recursive algorithm

DeepLearning.AI

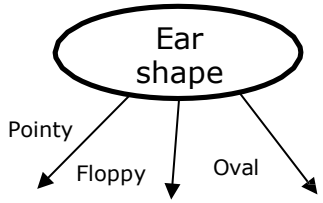
Stanford
ONLINE

Decision Tree Learning

Using one-hot encoding of categorical features

Features with three possible values

	Ear shape (x_1)	Face shape (x_2)	Whiskers (x_3)	Cat (y)
	Pointy	Round	Present	1
	Oval	Not round	Present	1
	Oval	Round	Absent	0
	Pointy	Not round	Present	0
	Oval	Round	Present	1
	Pointy	Round	Absent	1
	Floppy	Not round	Absent	0
	Oval	Round	Absent	1
	Floppy	Round	Absent	0
	Floppy	Round	Absent	0



3 possible values

One hot encoding

	~~Ear shape~~	Pointy ears	Floppy ears	Oval ears	Face shape	Whiskers	Cat
	~~Pointy~~	1	0	0	Round	Present	1
	~~Oval~~	0	0	1	Not round	Present	1
	~~Oval~~	0	0	1	Round	Absent	0
	~~Pointy~~	1	0	0	Not round	Present	0
	~~Oval~~	0	0	1	Round	Present	1
	~~Pointy~~	1	0	0	Round	Absent	1
	~~Floppy~~	0	1	0	Not round	Absent	0
	~~Oval~~	0	0	1	Round	Absent	1
	~~Floppy~~	0	1	0	Round	Absent	0
	~~Floppy~~	0	1	0	Round	Absent	0

One hot encoding

If a categorical feature can take on k values, create k binary features (0 or 1 valued).

One hot encoding

	~~Ear shape~~	Pointy ears	Floppy ears	Oval ears	Face shape	Whiskers	Cat
	~~Pointy~~	1	0	0	Round	Present	1
	~~Oval~~	0	0	1	Not round	Present	1
	~~Oval~~	0	0	1	Round	Absent	0
	~~Pointy~~	1	0	0	Not round	Present	0
	~~Oval~~	0	0	1	Round	Present	1
	~~Pointy~~	1	0	0	Round	Absent	1
	~~Floppy~~	0	1	0	Not round	Absent	0
	~~Oval~~	0	0	1	Round	Absent	1
	~~Floppy~~	0	1	0	Round	Absent	0
	~~Floppy~~	0	1	0	Round	Absent	0

One hot encoding and neural networks

	Pointy ears	Floppy ears	Round ears	Face shape	Whiskers	Cat
	1	0	0	~~Round~~ 1	~~Present~~ 1	1
	0	0	1	~~Not round~~ 0	~~Present~~ 1	1
	0	0	1	~~Round~~ 1	~~Absent~~ 0	0
	1	0	0	~~Not round~~ 0	~~Present~~ 1	0
	0	0	1	~~Round~~ 1	~~Present~~ 1	1
	1	0	0	~~Round~~ 1	~~Absent~~ 0	1
	0	1	0	~~Not round~~ 0	~~Absent~~ 0	1
	0	0	1	~~Round~~ 1	~~Absent~~ 0	1
	0	1	0	~~Round~~ 1	~~Absent~~ 0	1
	0	1	0	~~Round~~ 1	~~Absent~~ 0	1

DeepLearning.AI

Stanford
ONLINE

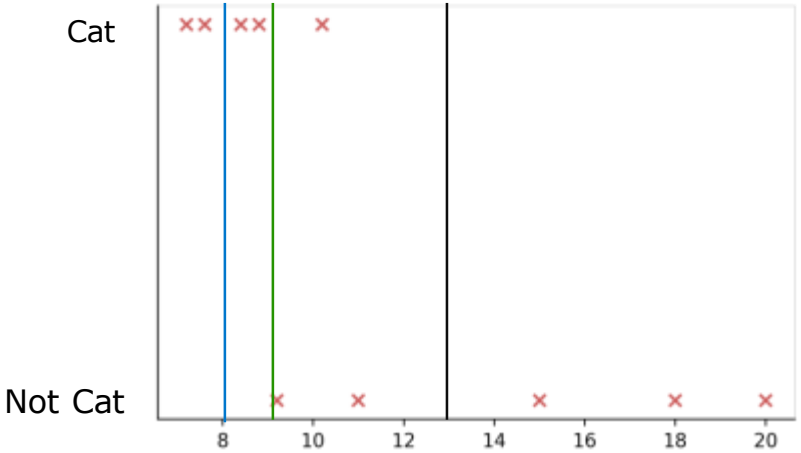
Decision Tree Learning

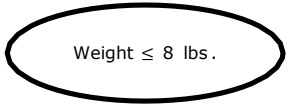
Continuous valued features

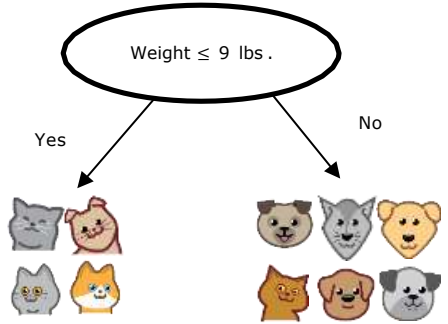
Continuous features

	Ear shape	Face shape	Whiskers	Weight (lbs.)	Cat
	Pointy	Round	Present	7.2	1
	Floppy	Not round	Present	8.8	1
	Floppy	Round	Absent	15	0
	Pointy	Not round	Present	9.2	0
	Pointy	Round	Present	8.4	1
	Pointy	Round	Absent	7.6	1
	Floppy	Not round	Absent	11	0
	Pointy	Round	Absent	10.2	1
	Floppy	Round	Absent	18	0
	Floppy	Round	Absent	20	0

Splitting on a continuous variable







$$\mathrm{H}(0.5) - \left(\frac{2}{10} H\left(\frac{2}{2}\right) + \frac{8}{10} H\left(\frac{3}{8}\right)\right) = 0.24$$

$$\mathrm{H}(0.5) - \left(\frac{4}{10} H\left(\frac{4}{4}\right) + \frac{6}{10} H\left(\frac{1}{6}\right)\right) = 0.61$$

$$\mathrm{H}(0.5) - \left(\frac{7}{10} H\left(\frac{5}{7}\right) + \frac{3}{10} H\left(\frac{0}{3}\right)\right) = 0.40$$

DeepLearning.AI

Stanford
ONLINE

Decision Tree Learning

Regression Trees (optional)

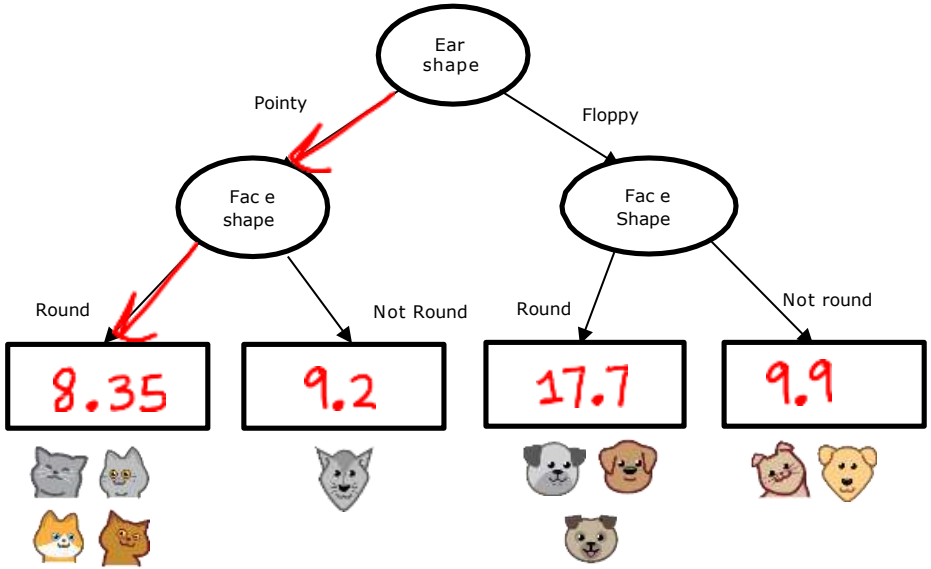
Regression with Decision Trees: Predicting a number

	Ear shape	Face shape	Whiskers	Weight (lbs.)
	Pointy	Round	Present	7.2
	Floppy	Not round	Present	8.8
	Floppy	Round	Absent	15
	Pointy	Not round	Present	9.2
	Pointy	Round	Present	8.4
	Pointy	Round	Absent	7.6
	Floppy	Not round	Absent	11
	Pointy	Round	Absent	10.2
	Floppy	Round	Absent	18
	Floppy	Round	Absent	20

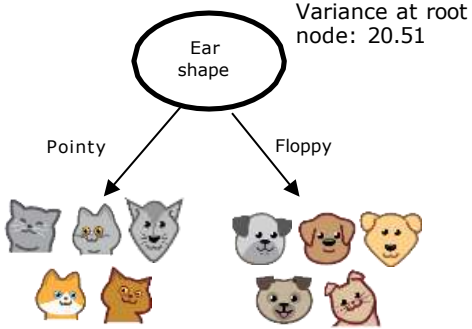
X

y

Regression with Decision Trees



Choosing a split



Variance at root node: 20.51

Weights: 7.2, 9.2, 8.4,7.6, 10.2

Variance: 1.47

Weights: 8.8, 15, 11, 18, 20

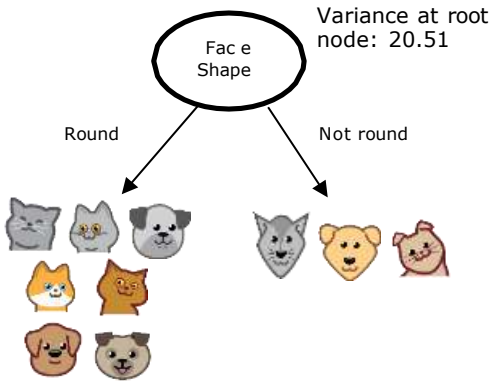
Variance: 21.87

$w^{\text{left}} = 5/10$

$w^{\text{right}} = 5/10$

$$20.51 - \left(\frac{5}{10} * 1.47 + \frac{5}{10} * 21.87\right)$$

$$= 8.84$$



Variance at root node: 20.51

Weights: 7.2, 15, 8.4, 7.6,10.2, 18, 20

Variance: 27.80

Weights: 8.8,9.2,11

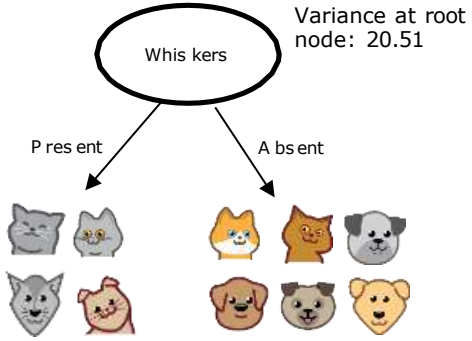
Variance: 1.37

$w^{\text{left}} = 7/10$

$w^{\text{right}} = 3/10$

$$20.51 - \left(\frac{7}{10} * 27.80 + \frac{3}{10} * 1.37\right)$$

$$= 0.64$$



Variance at root node: 20.51

Weights: 7.2, 8.8, 9.2, 8.4

Variance: 0.75

Weights: 15, 7.6, 11, 10.2, 18, 20

Variance: 23.32

$w^{\text{left}} = 4/10$

$w^{\text{right}} = 6/10$

$$20.51 - \left(\frac{4}{10} * 0.75 + \frac{6}{10} * 23.32\right)$$

$$= 6.22$$

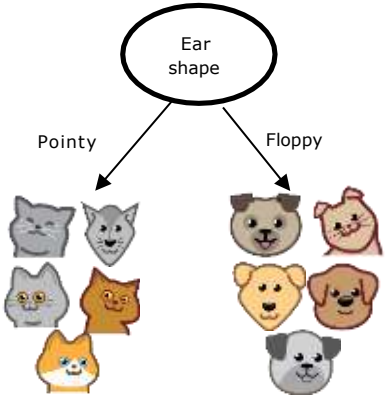
DeepLearning.AI

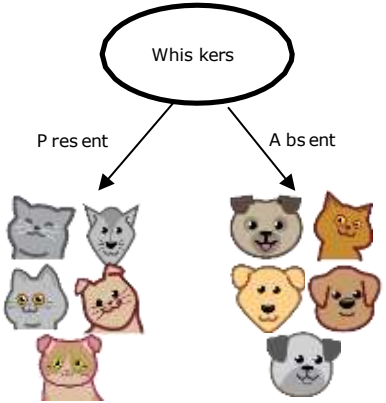
Stanford
ONLINE

Tree ensembles

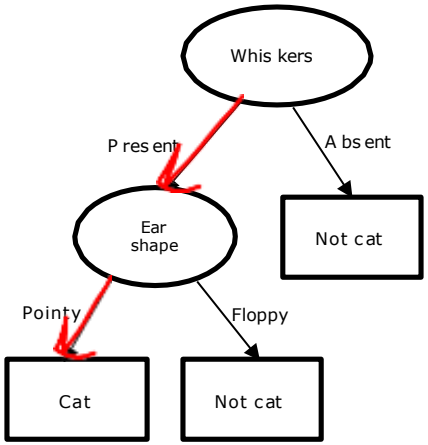
Using multiple decision trees

Trees are highly sensitive to small changes of the data

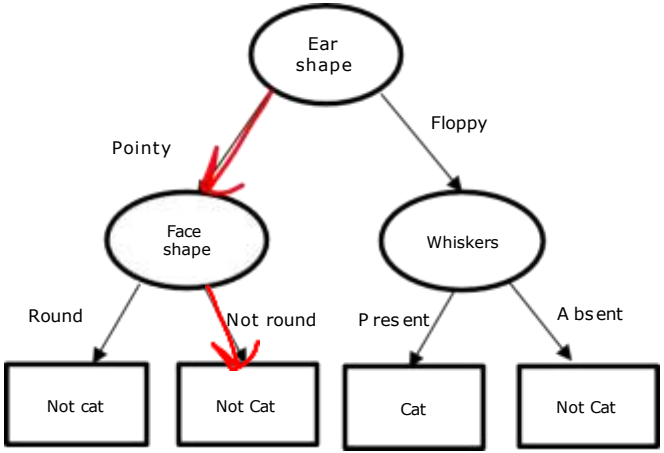




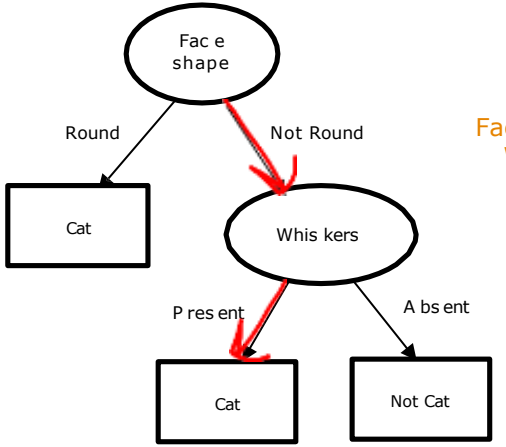
Tree ensemble



Prediction: Cat



Prediction: Not cat



Prediction: Cat

New test example

Ear shape: Pointy
Face shape: Not Round
Whiskers: Present

Final prediction: Cat

DeepLearning.AI

Stanford
ONLINE

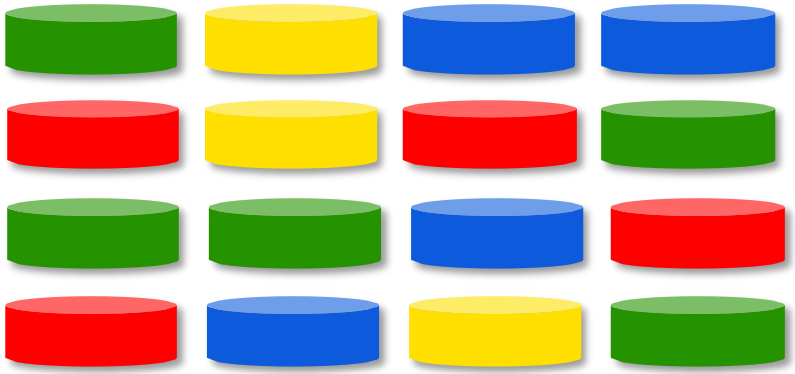
Tree ensembles

Sampling with replacement

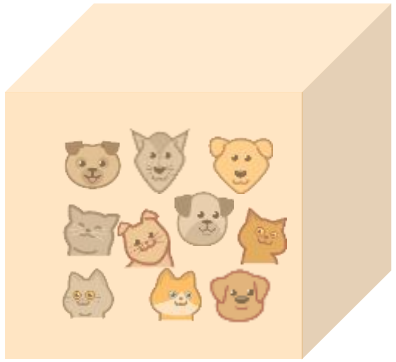
Sampling with replacement

Tokens

Sampling with replacement:



Sampling with replacement



	Ear shape	Face shape	Whiskers	Cat
	Pointy	Round	Present	1
	Floppy	Not round	Absent	0
	Pointy	Round	Absent	1
	Pointy	Not round	Present	0
	Floppy	Not round	Absent	0
	Pointy	Round	Absent	1
	Pointy	Round	Present	1
	Floppy	Not round	Present	1
	Floppy	Round	Absent	0
	Pointy	Round	Absent	1

DeepLearning.AI

Stanford
ONLINE

Tree ensembles

Random forest algorithm

Generating a tree sample

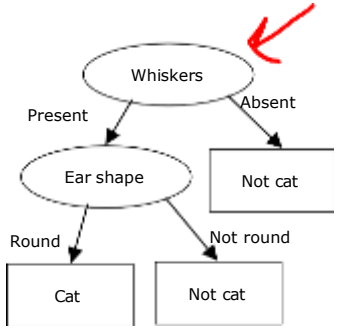
Given training set of size m

For $b = 1$ to B:

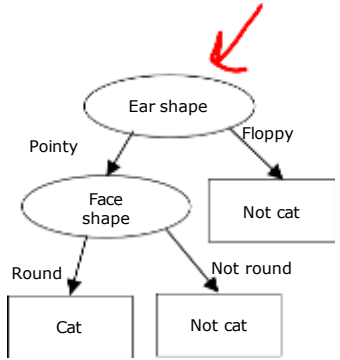
Use sampling with replacement to create a new training set of size m

Train a decision tree on the new dataset

Ear shape	Face shape	Whiskers	Cat
Pointy	Round	Present	Yes
Floppy	Round	Absent	No
Floppy	Round	Absent	No
Pointy	Round	Present	Yes
Pointy	Not Round	Present	Yes
Floppy	Round	Absent	No
Floppy	Round	Present	Yes
Pointy	Not Round	Absent	No
Pointy	Not Round	Absent	No
Pointy	Not Round	Present	Yes



Ear shape	Face shape	Whiskers	Cat
Pointy	Round	Present	Yes
Pointy	Round	Absent	Yes
Floppy	Not Round	Absent	No
Floppy	Not Round	Absent	No
Pointy	Round	Absent	Yes
Floppy	Round	Absent	No
Floppy	Round	Absent	No
Floppy	Round	Absent	No
Pointy	Not Round	Absent	No
Pointy	Round	Present	Yes



...

Bagged decision tree

Randomizing the feature choice

At each node, when choosing a feature to use to split, if n features are available, pick a random subset of $k < n$ features and allow the algorithm to only choose from that subset of features.

$k = \sqrt{n}$

Random forest algorithm

DeepLearning.AI

Stanford
ONLINE

Tree ensembles

XGBoost

Boosted trees intuition

Given training set of size m

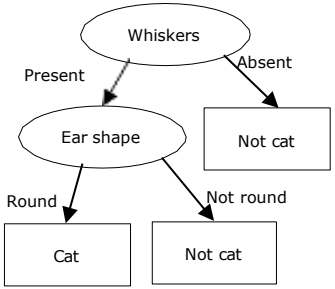
For b = 1 to B:

Use sampling with replacement to create a new training set of size m

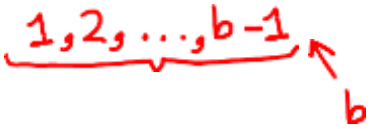
But instead of picking from all examples with equal (1/m) probability, make it more likely to pick examples that the previously trained trees misclassify

Train a decision tree on the new dataset

Ear shape	Face shape	Whiskers	Cat
Pointy	Round	Present	Yes
Floppy	Round	Absent	No
Floppy	Round	Absent	No
Pointy	Round	Present	Yes
Pointy	Not Round	Present	Yes
Floppy	Round	Absent	No
Floppy	Round	Present	Yes
Pointy	Not Round	Absent	No
Pointy	Not Round	Absent	No
Pointy	Not Round	Present	Yes



Ear shape	Face shape	Whiskers	Prediction
Pointy	Round	Present	Cat ✓
Floppy	Not Round	Present	Not cat ✗
Floppy	Round	Absent	Not cat ✓
Pointy	Not Round	Present	Not cat ✓
Pointy	Round	Present	Cat ✓
Pointy	Round	Absent	Not cat ✗
Floppy	Not Round	Absent	Not cat ✓
Pointy	Round	Absent	Not cat ✗
Floppy	Round	Absent	Not cat ✓
Floppy	Round	Absent	Not cat ✓



XGBoost (eXtreme Gradient Boosting)

- Open source implementation of boosted trees
- Fast efficient implementation
- Good choice of default splitting criteria and criteria for when to stop splitting
- Built in regularization to prevent overfitting
- Highly competitive algorithm for machine learning competitions (eg: Kaggle competitions)

Using XGBoost

Classification

```
from xgboost import XGBClassifier

model = XGBClassifier()

model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Regression

```
from xgboost import XGBRegressor

model = XGBRegressor()

model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

DeepLearning.AI

Stanford
ONLINE

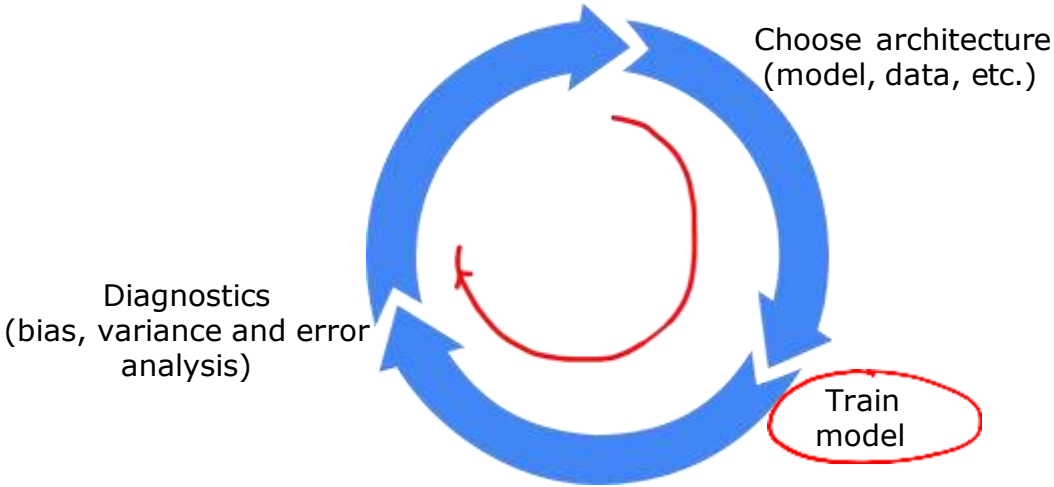
Conclusion

When to use decision trees

Decision Trees vs Neural Networks

Decision Trees and Tree ensembles

- Works well on tabular (structured) data
- Not recommended for unstructured data (images, audio, text)
- Fast



Decision Trees vs Neural Networks

Decision Trees and Tree ensembles

- Works well on tabular (structured) data
- Not recommended for unstructured data (images, audio, text)
- Fast
- Small decision trees may be human interpretable

Neural Networks

- Works well on all types of data, including tabular (structured) and unstructured data
- May be slower than a decision tree
- Works with transfer learning
- When building a system of multiple models working together, it might be easier to string together multiple neural networks