Part 2: Advanced Learning Algorithms

Week 4+5: Neural Networks

Contents

Contents

Contents

5. Vectorization
    - How NNs are implemented efficiently
    - Matrix multiplication
    - Matrix multiplication rules
    - Matrix multiplication code

1. Neural Network Intuition

Overview

Advanced learning algorithms

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

Neurons and Brain

Neural networks

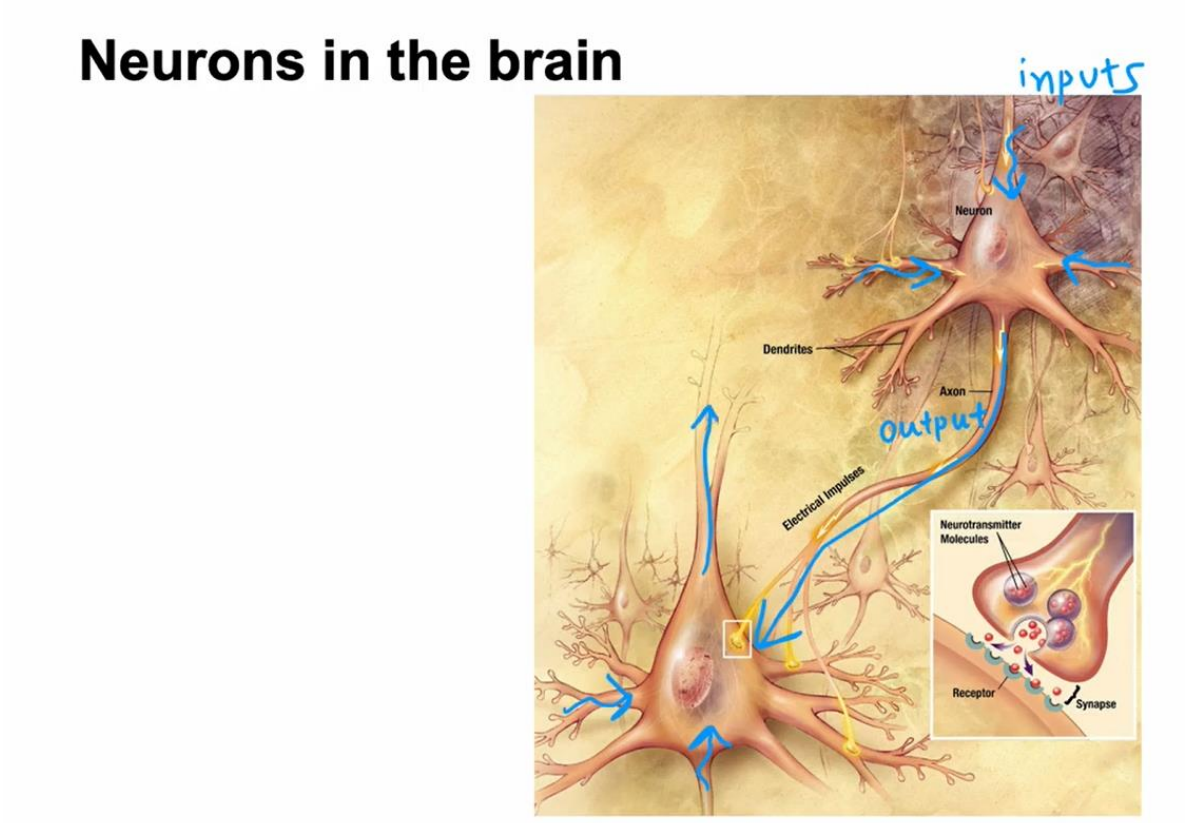
Origins: Algorithms that try to mimic the brain.

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 and Brain



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

Neurons and Brain

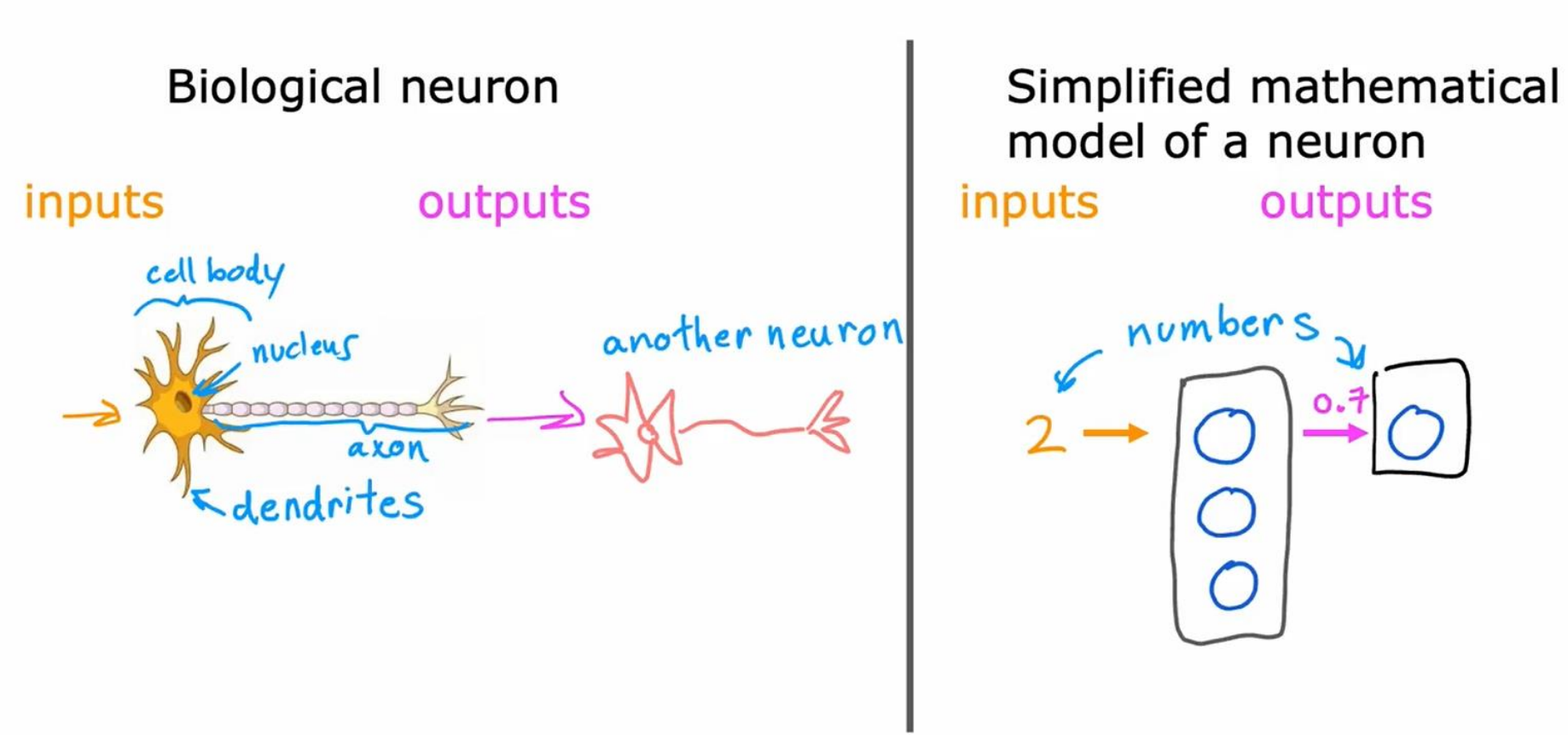
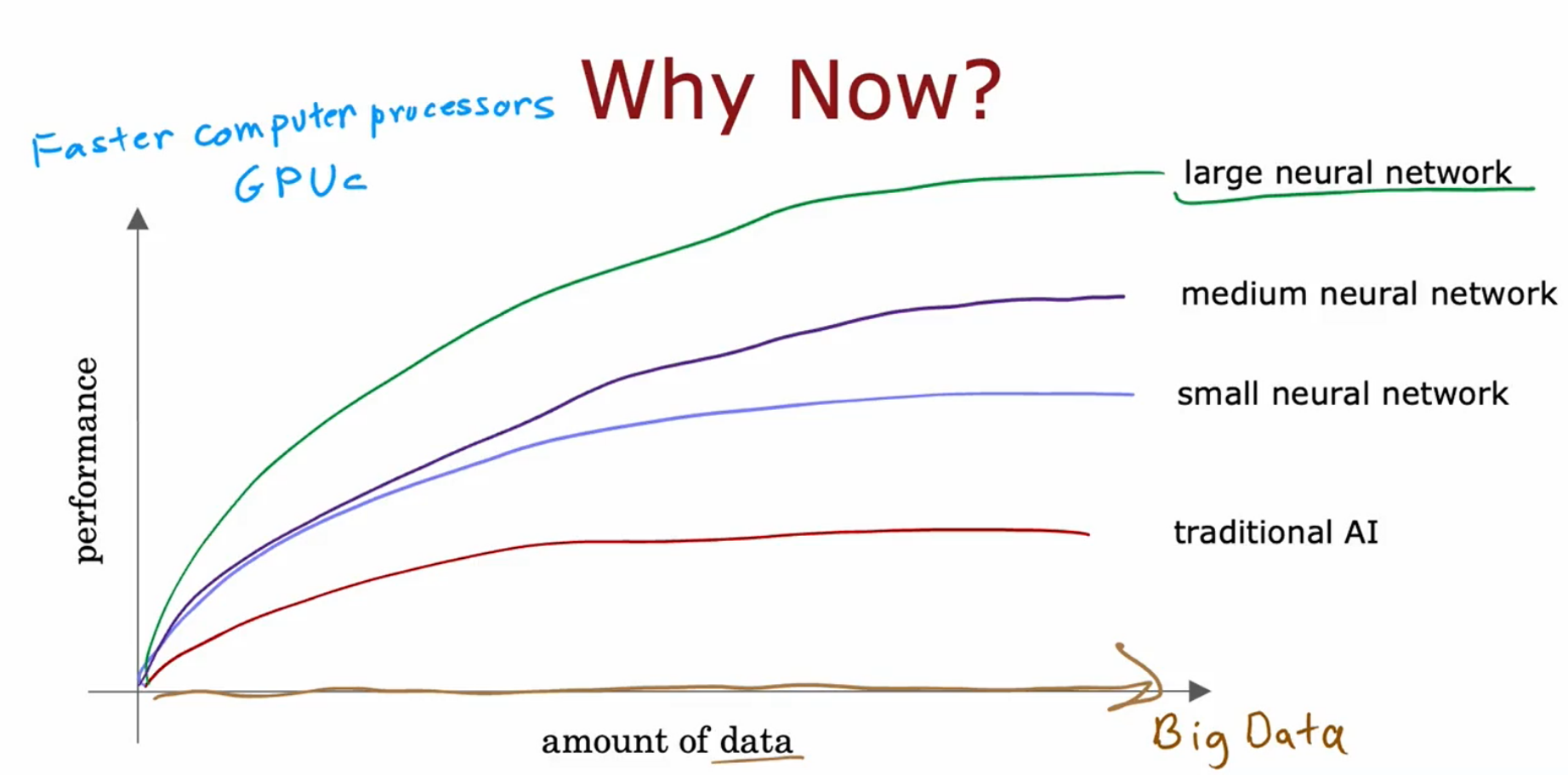
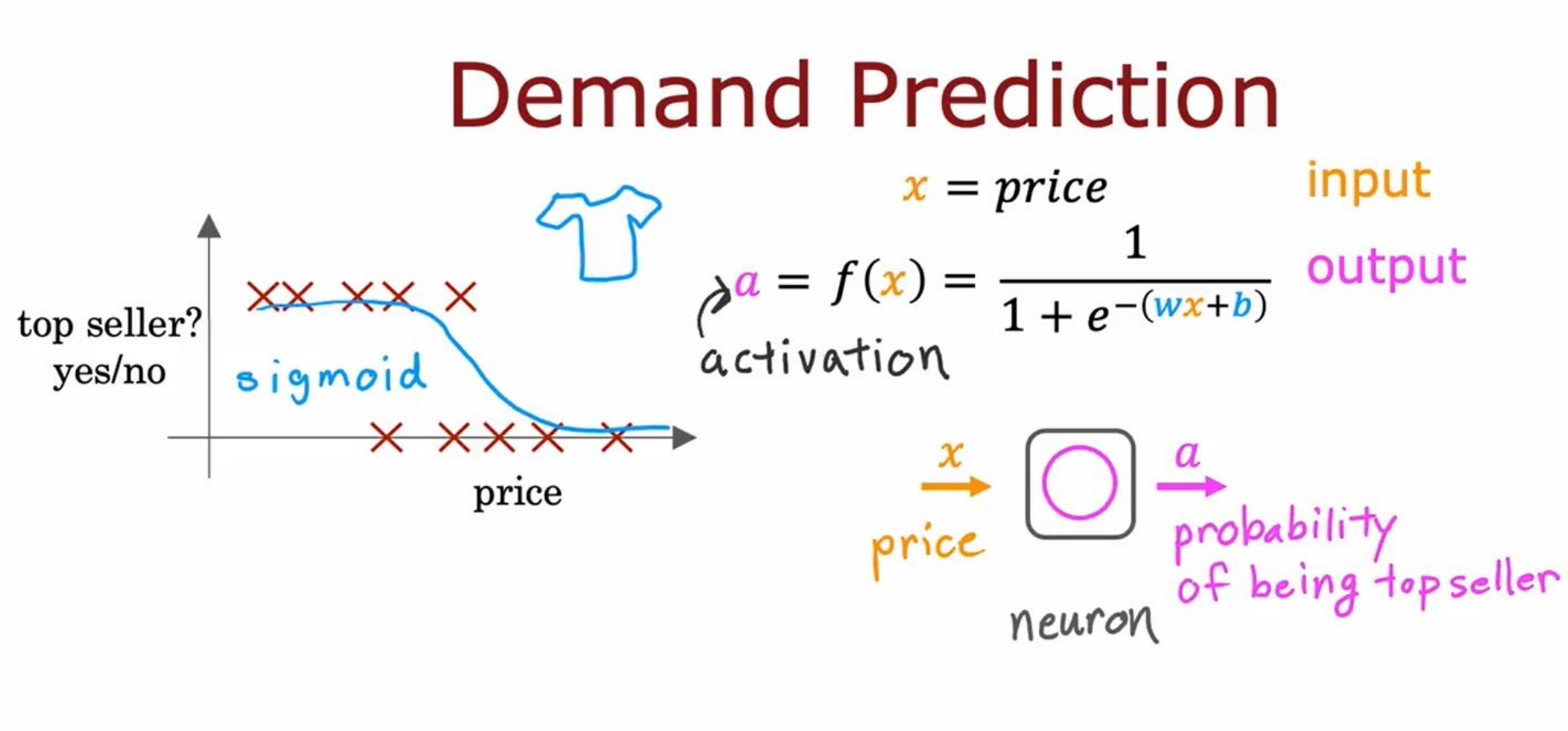


image source: https://biologydictionary.net/sensory-neuron/

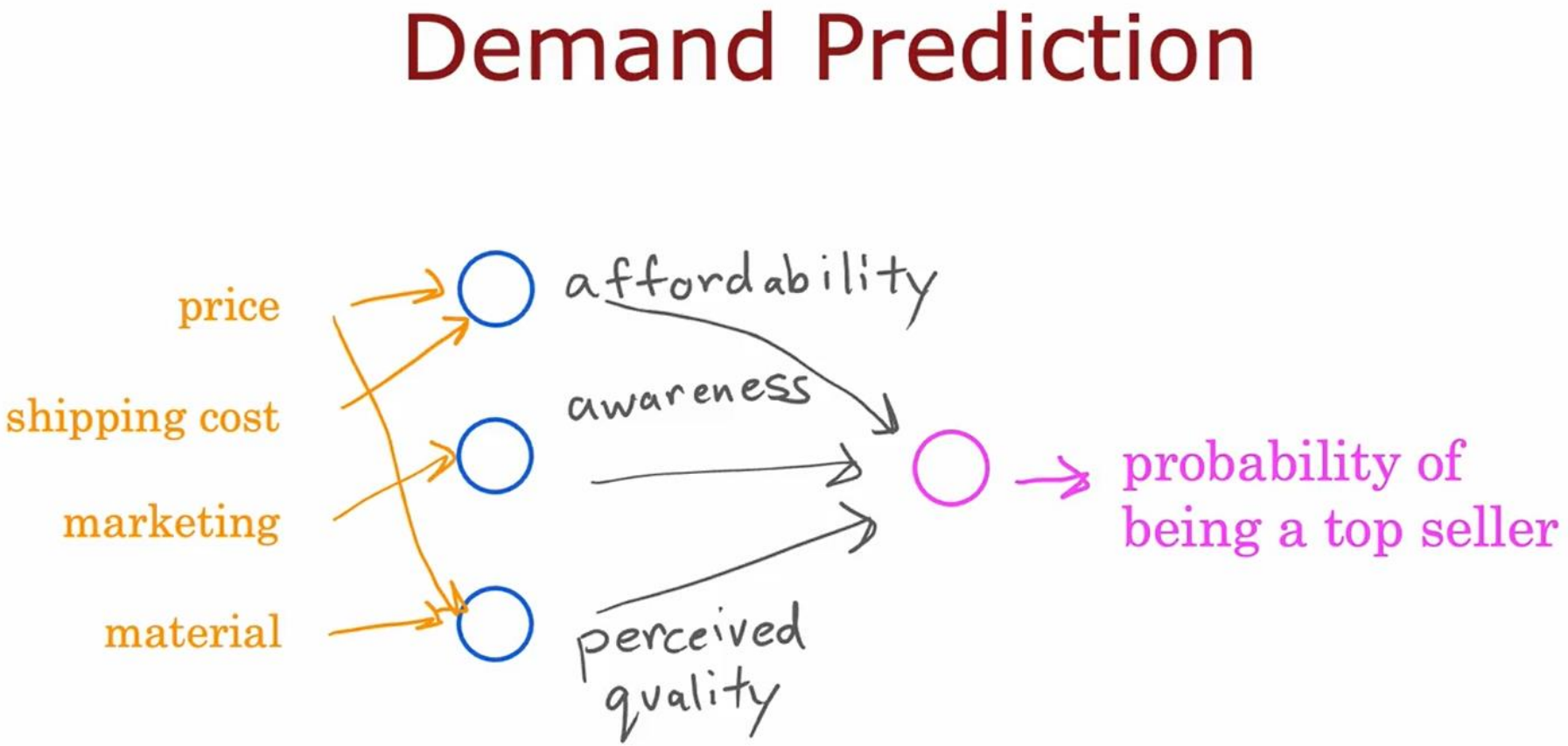
Neurons and Brain



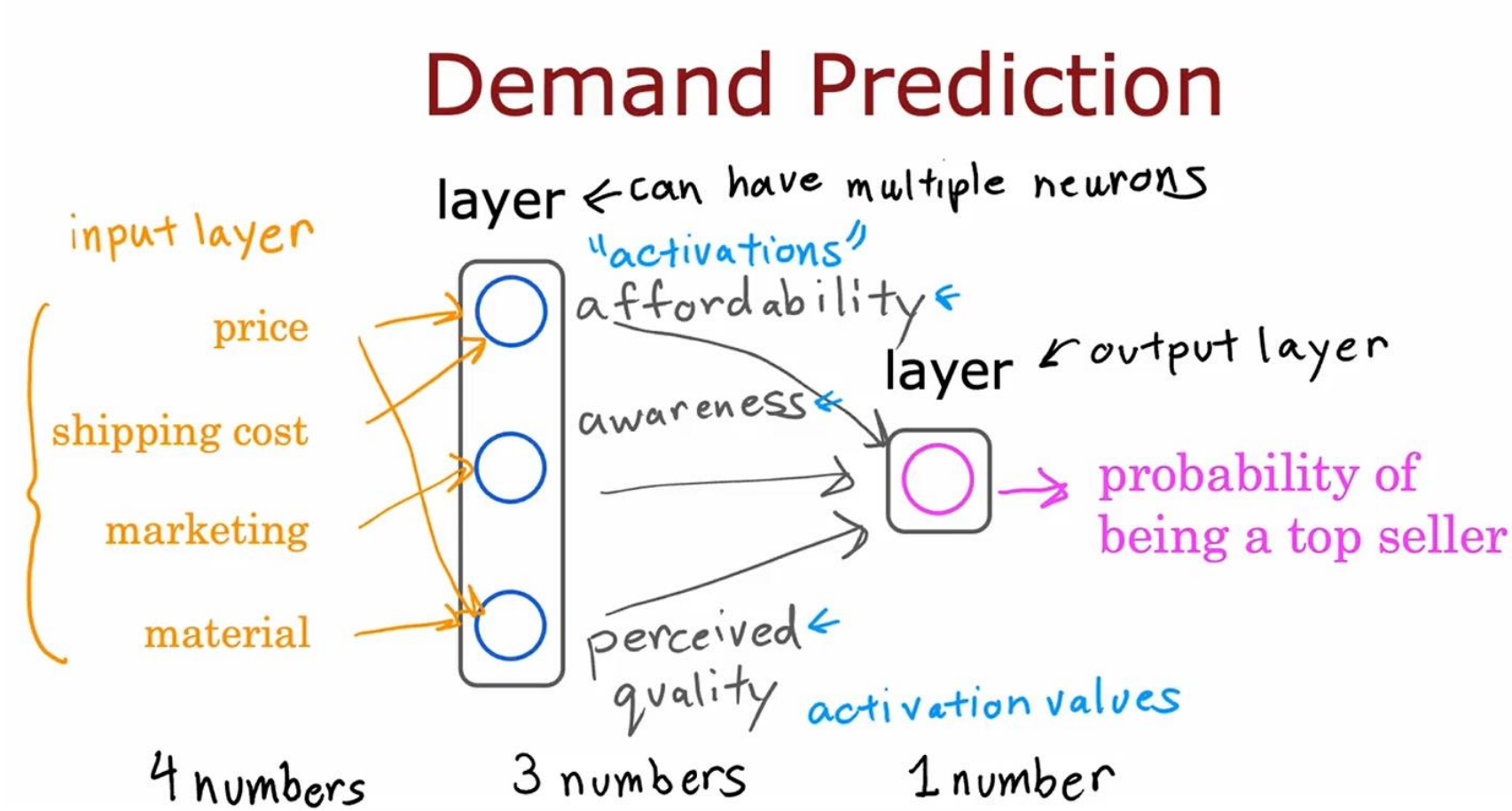
Demand Prediction



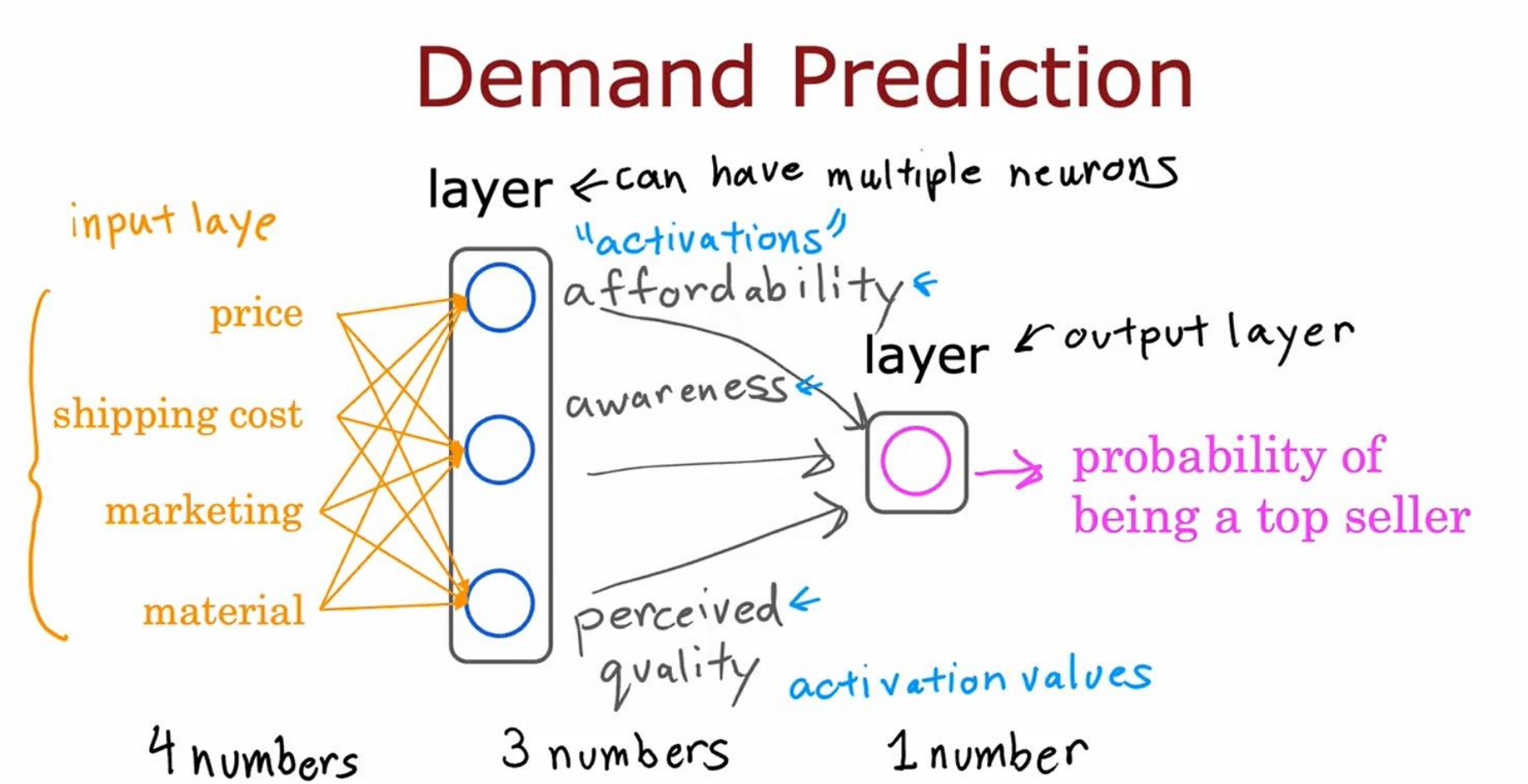
Demand Prediction



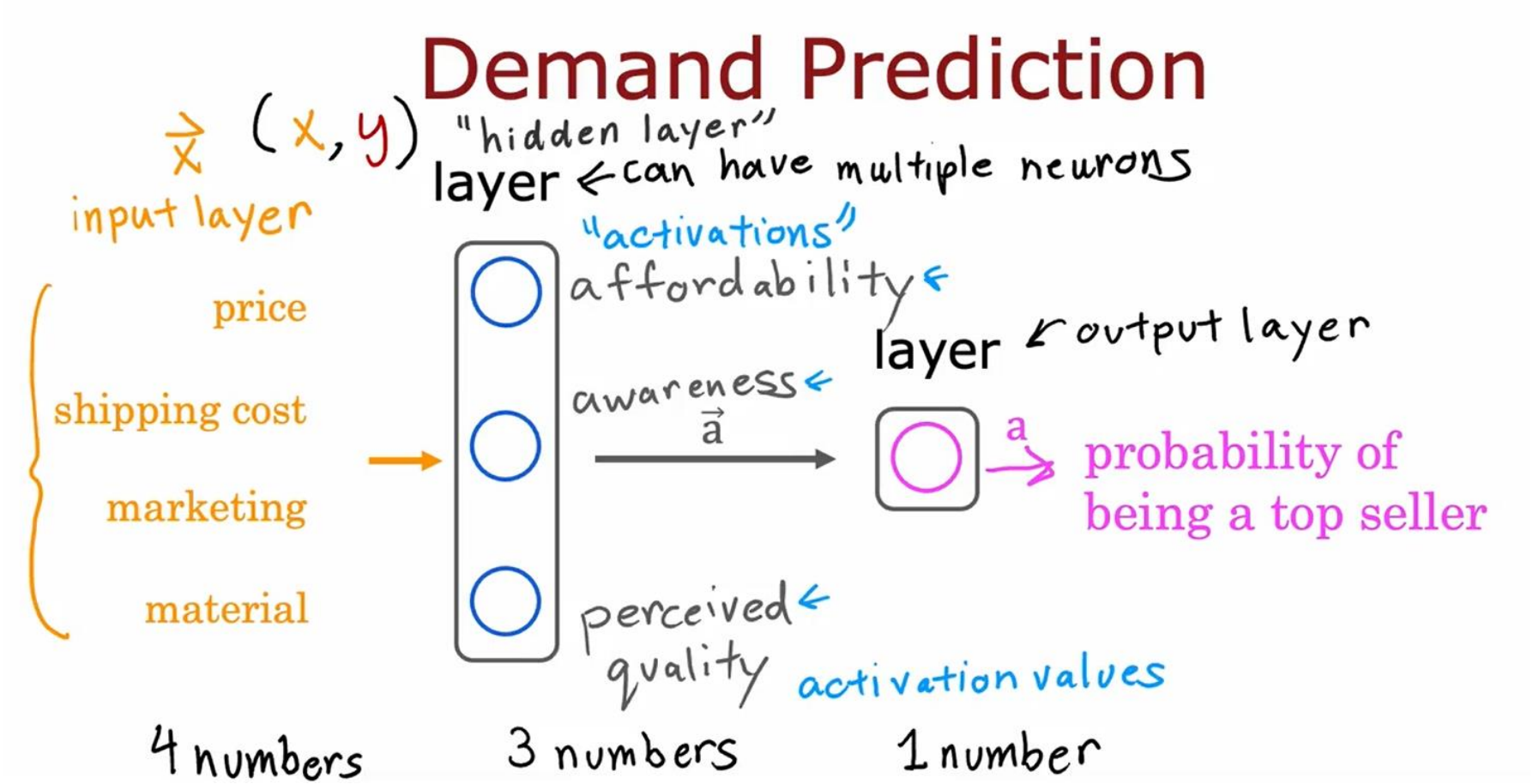
Demand Prediction



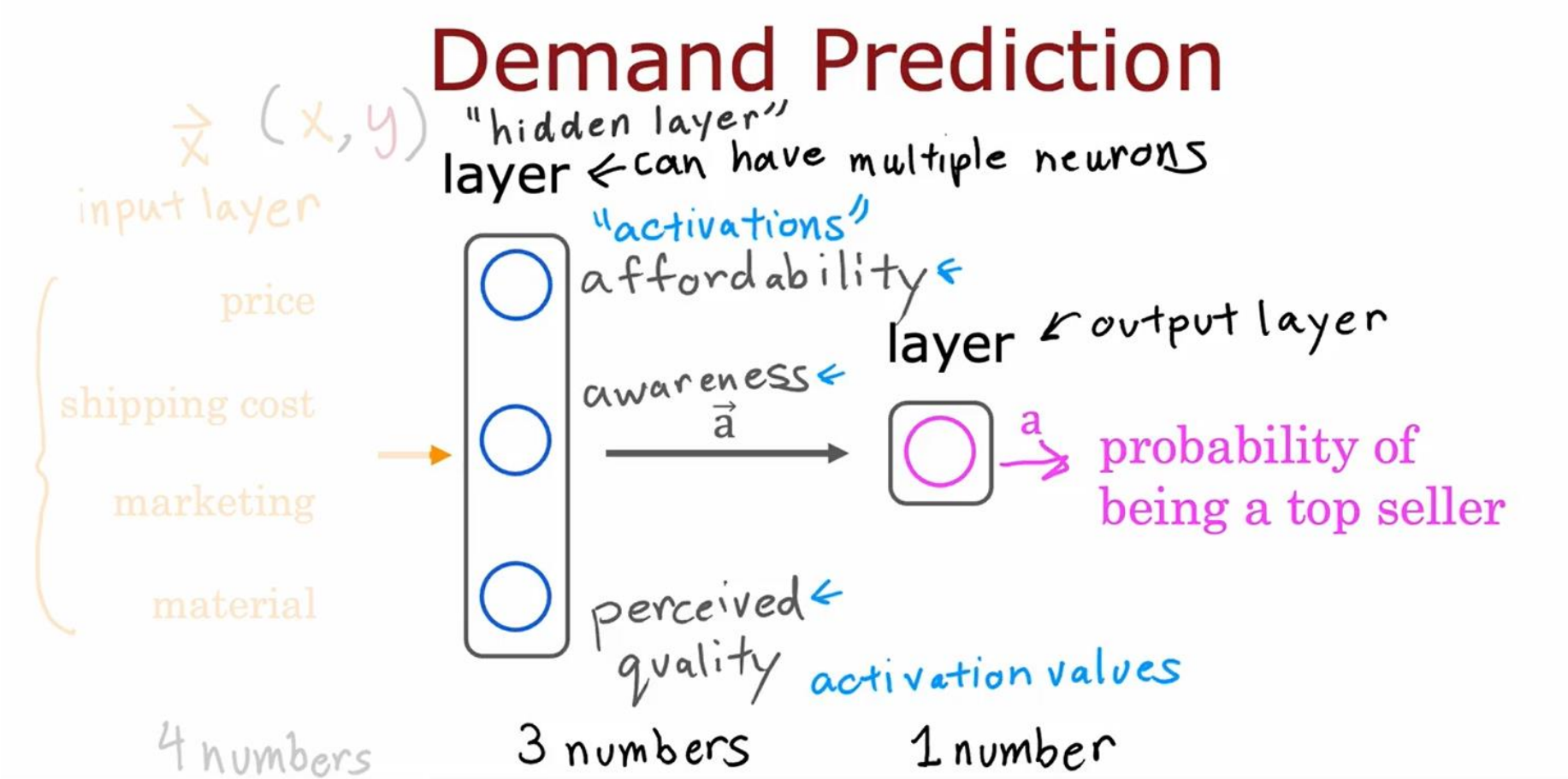
Demand Prediction



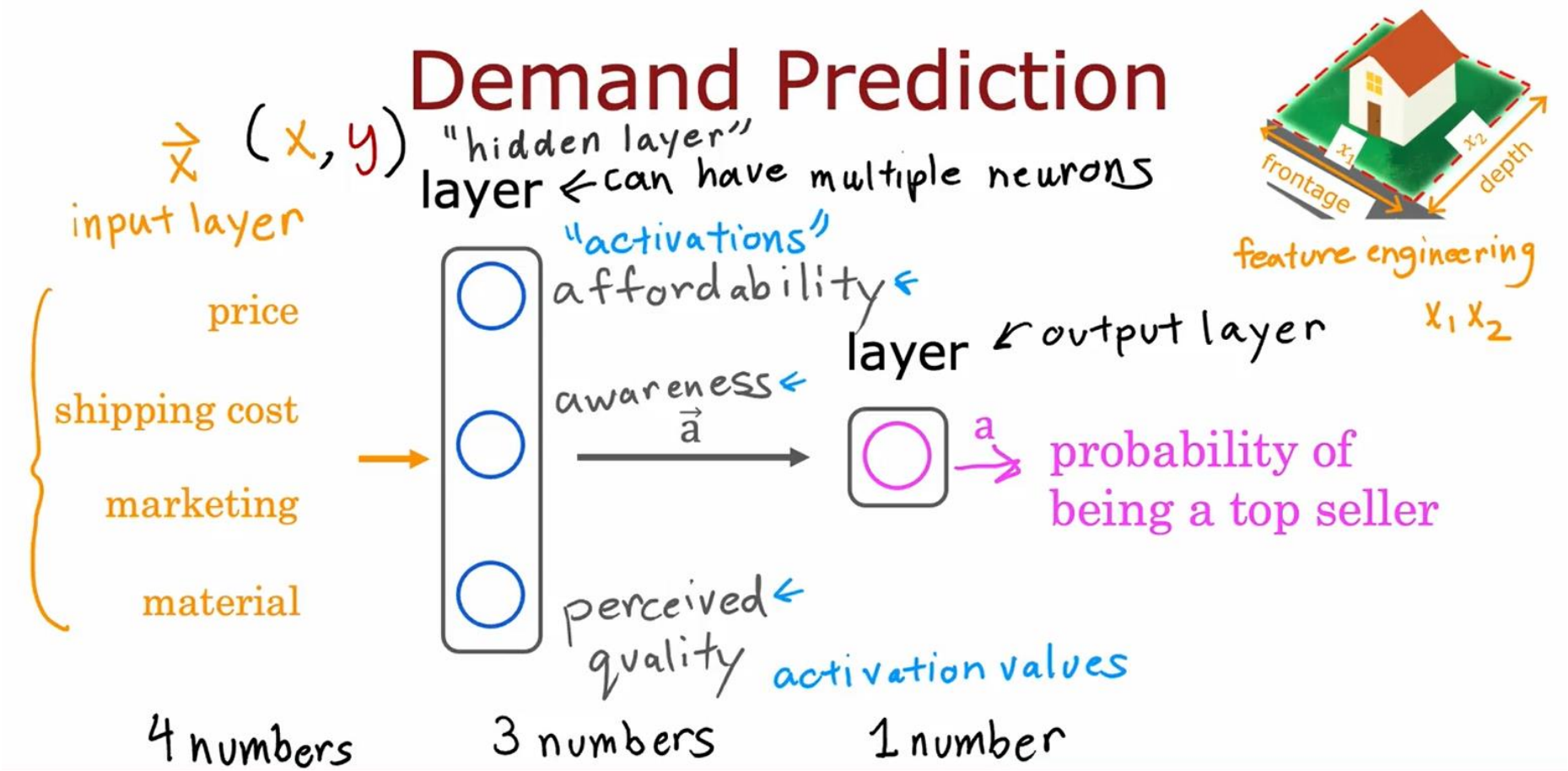
Demand Prediction



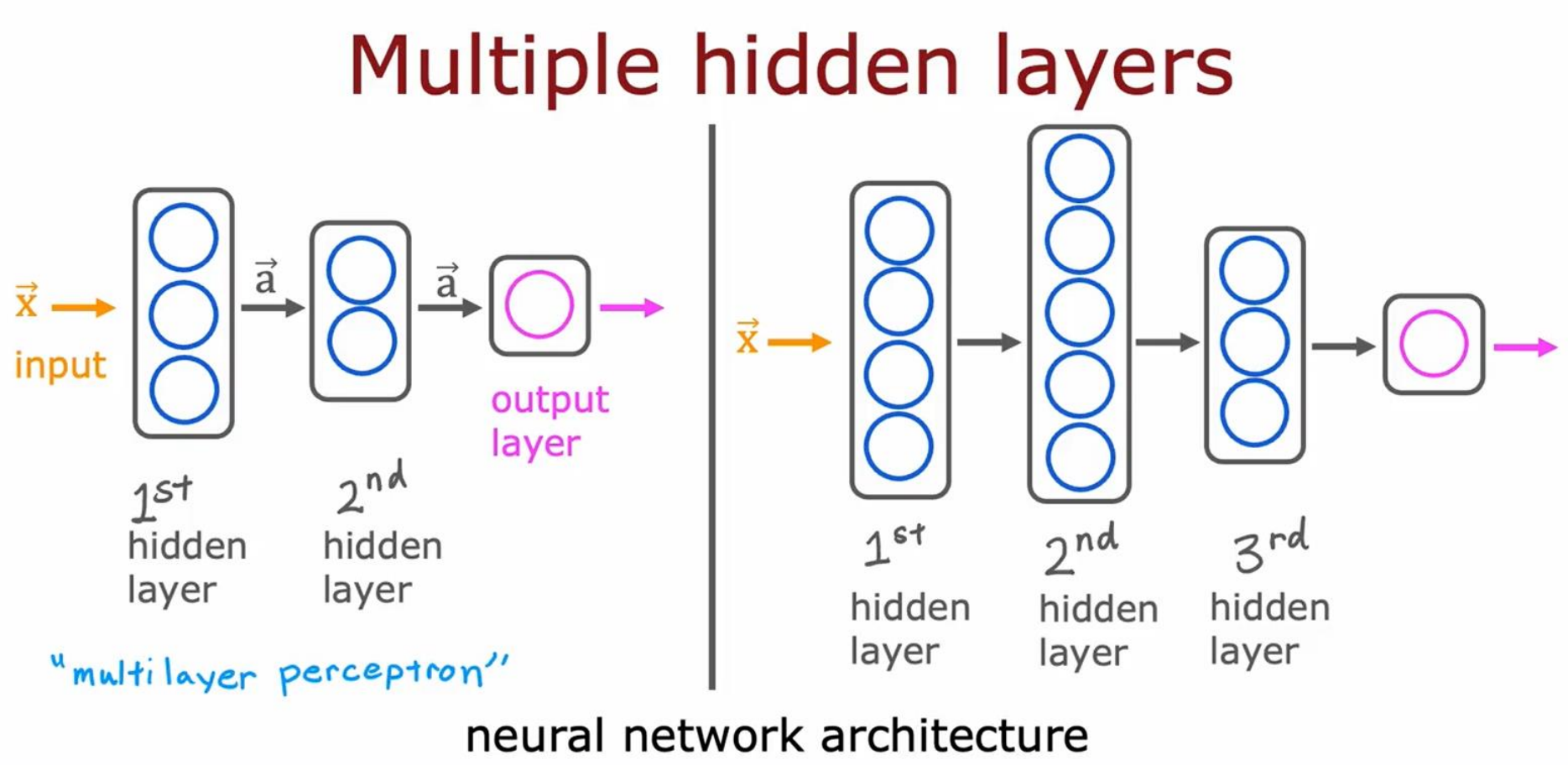
Demand Prediction



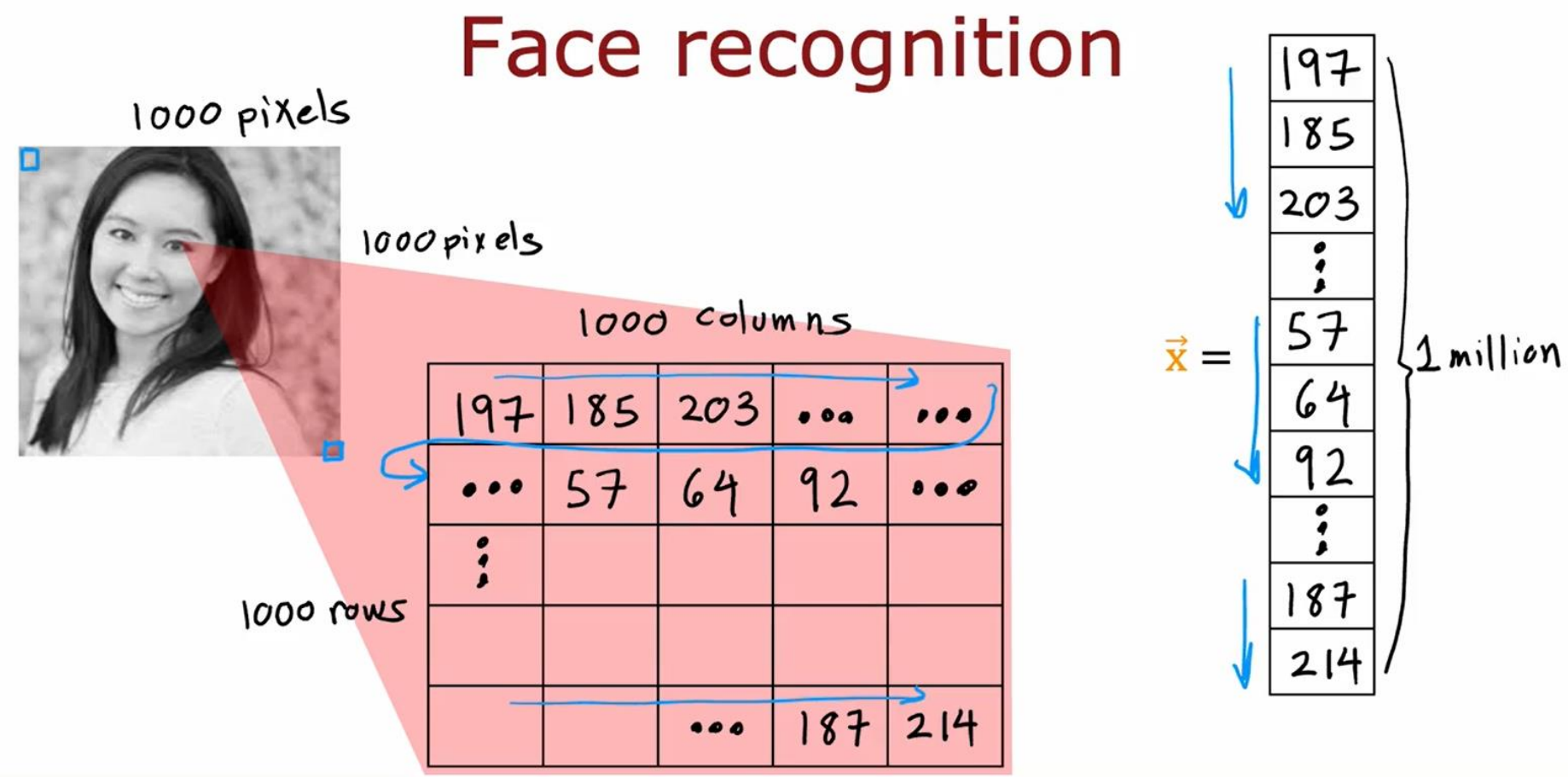
Demand Prediction



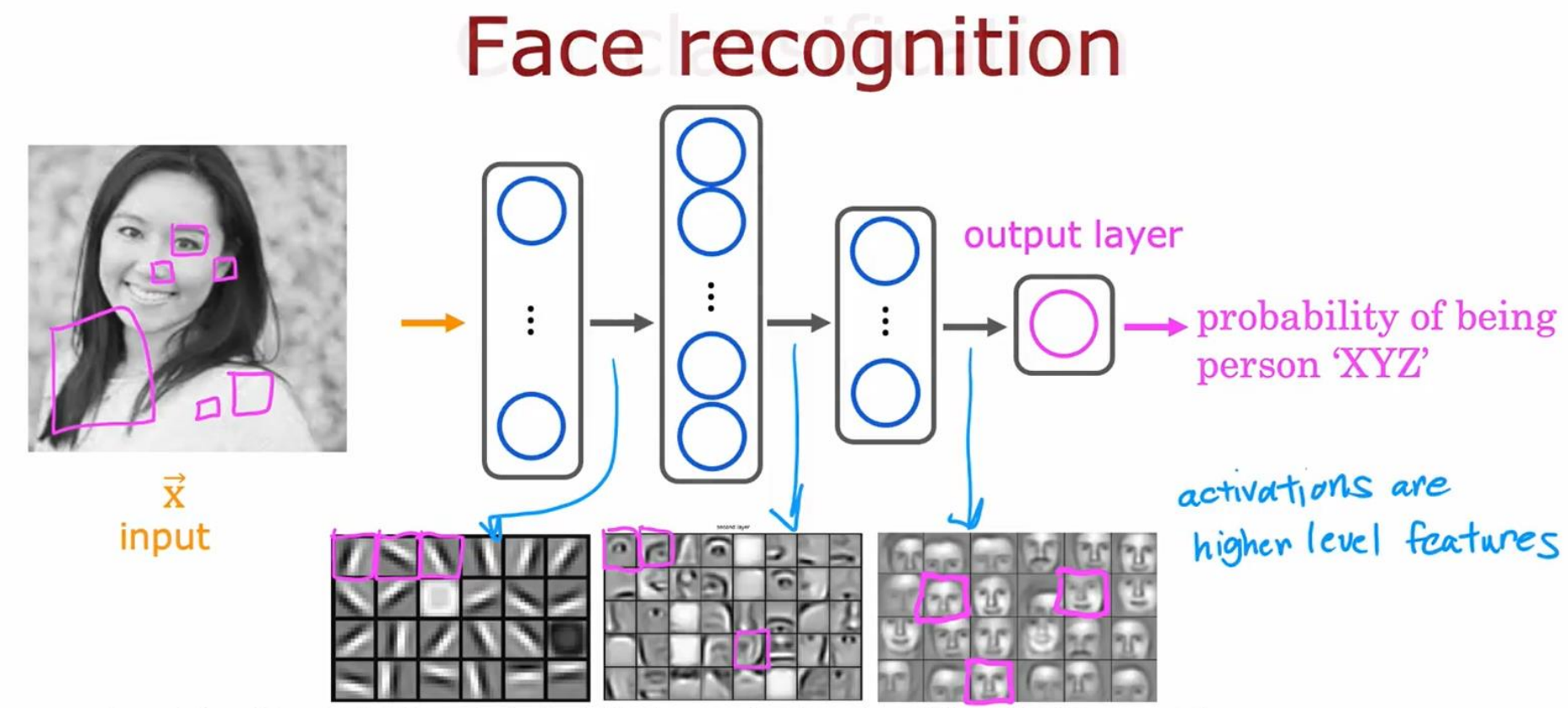
Demand Prediction



Recognizing Images

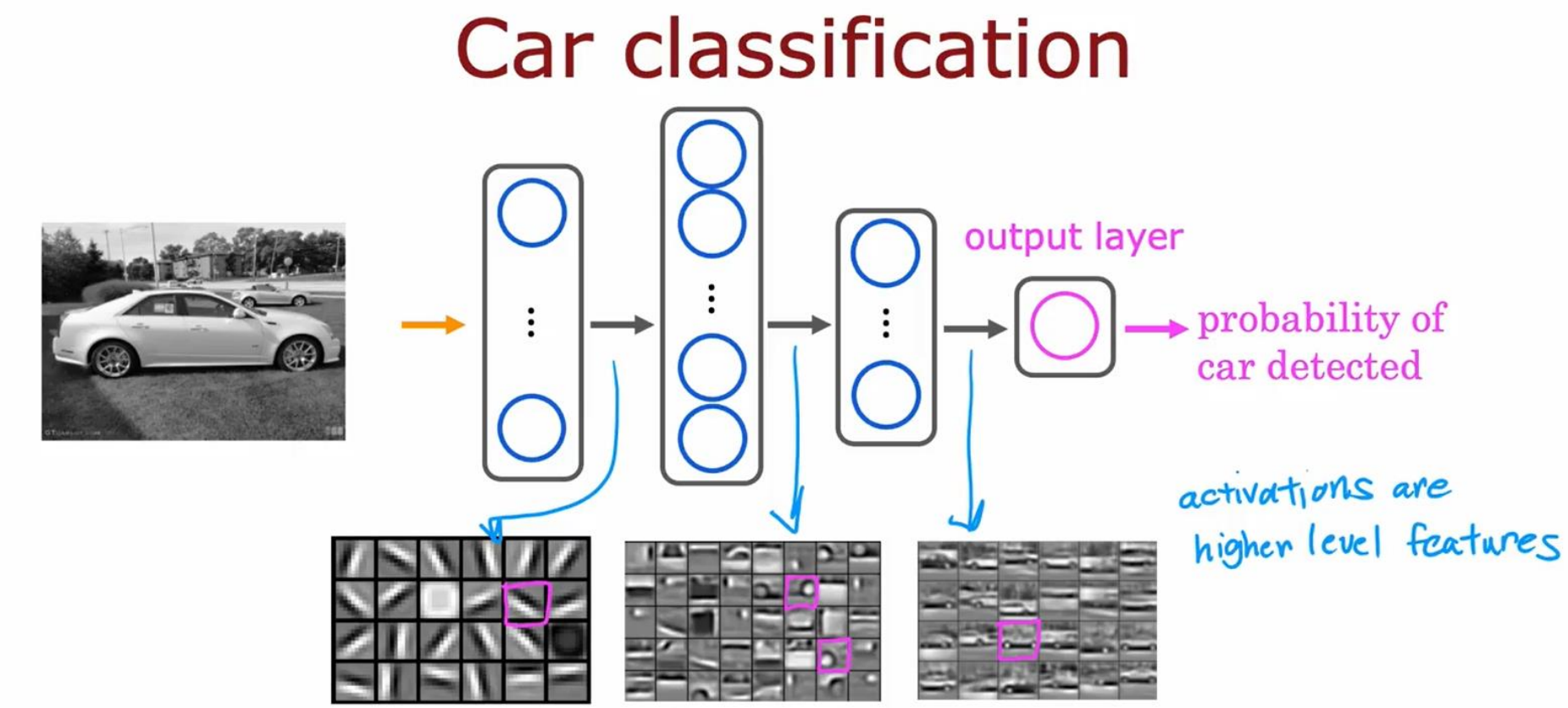


Recognizing Images



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

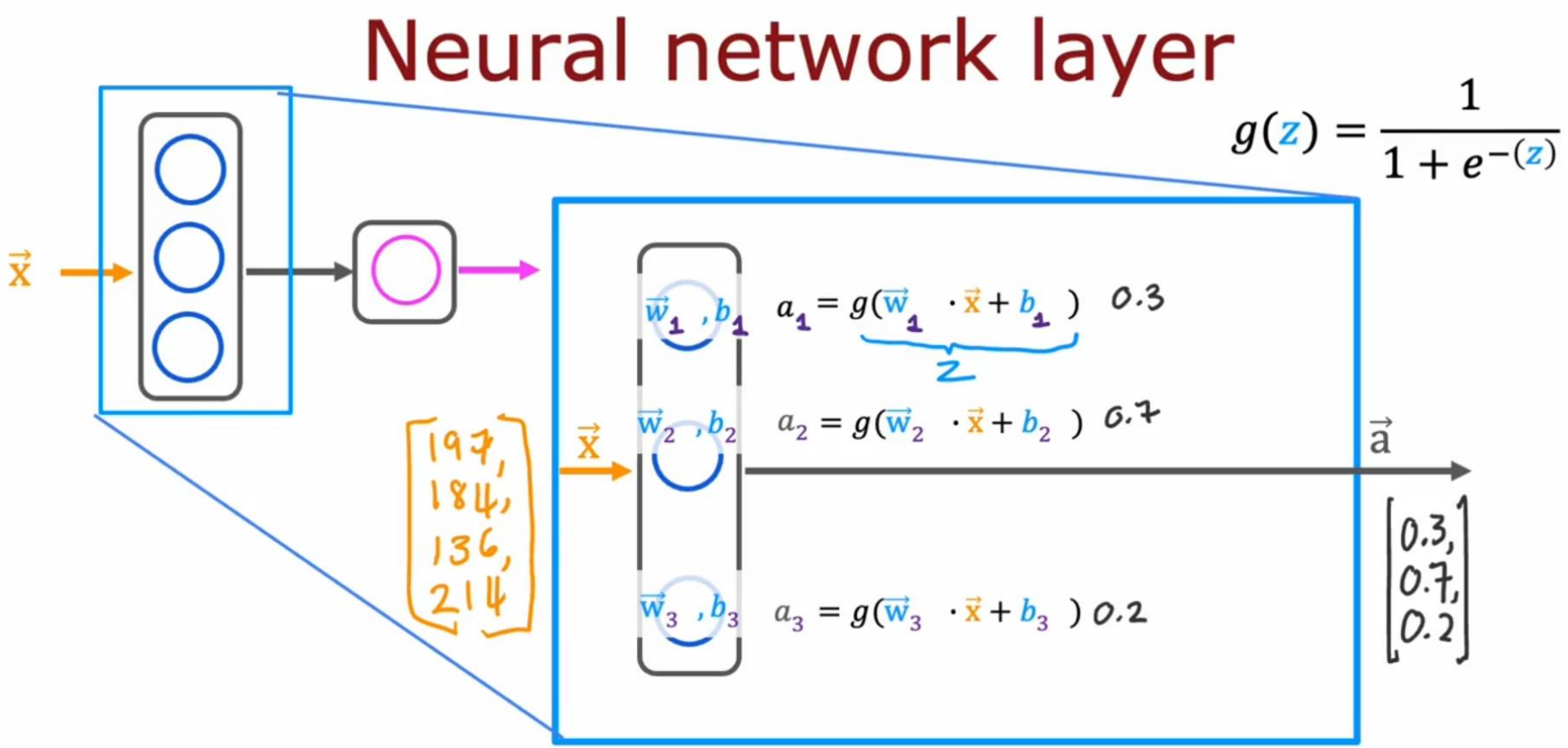
Recognizing Images



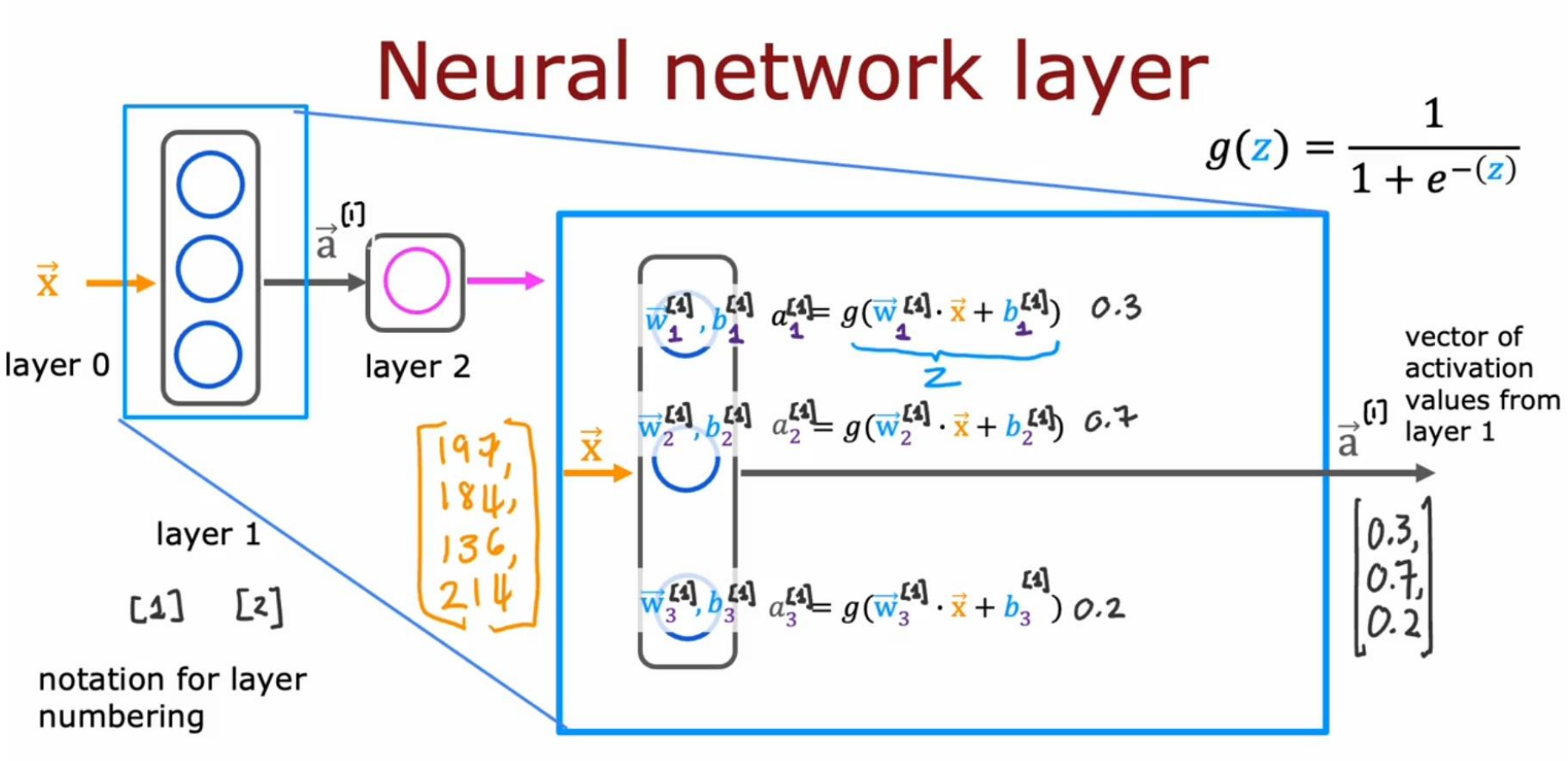
source: Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations by Honglak Lee, Roger Grosse, Ranganath Andrew Y. Ng

2. Neural Network Model

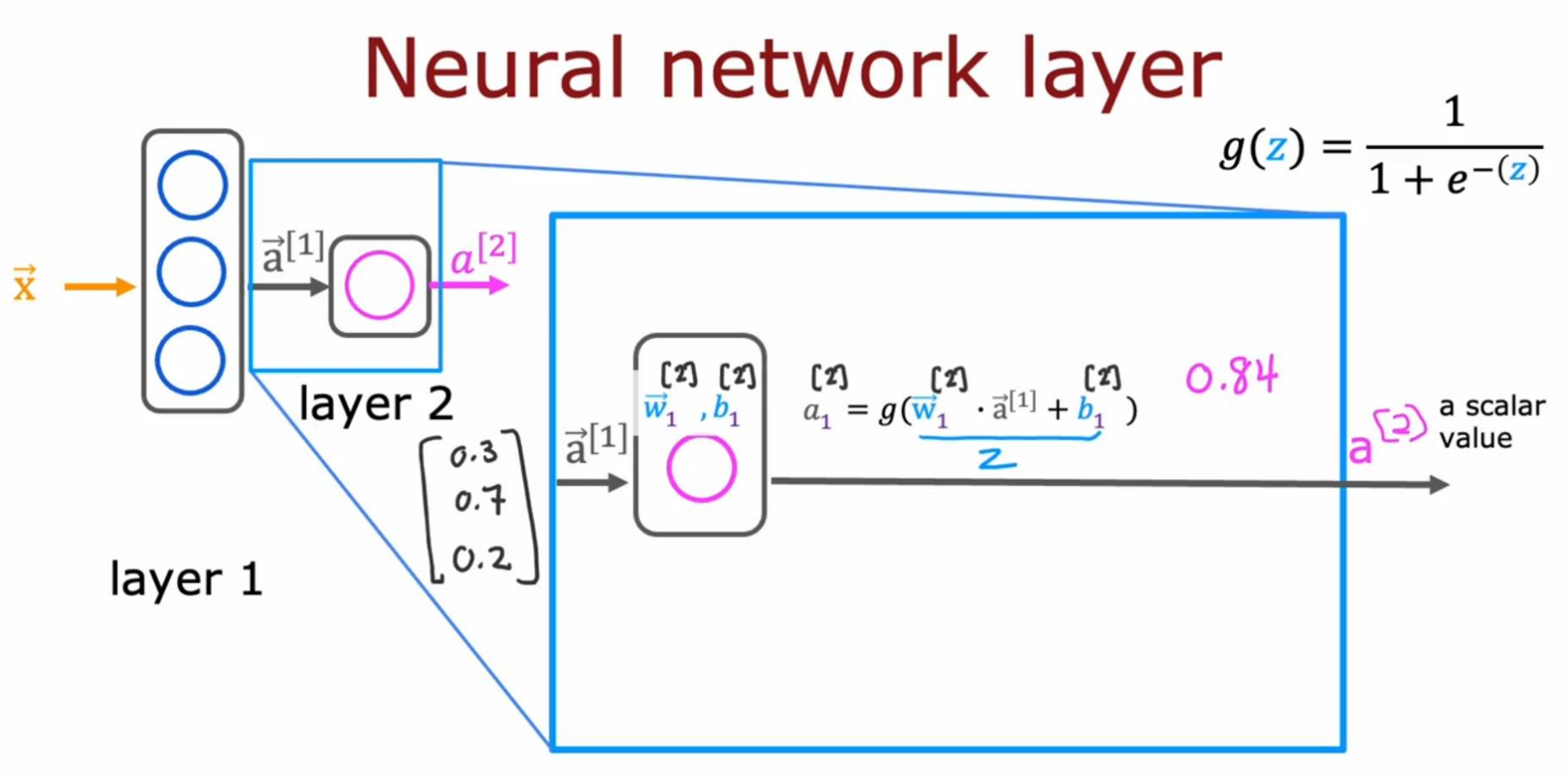
NN Layer


Neural network layer
$g(z) = \frac{1}{1+e^{-(z)}}$
$\vec{x}$
$\vec{w}_1, b_1$
$a_1 = g(\vec{w}_1 \cdot \vec{x} + b_1)$ 0.3
z
$\vec{w}_2, b_2$
$a_2 = g(\vec{w}_2 \cdot \vec{x} + b_2)$ 0.7
$\vec{w}_3, b_3$
$a_3 = g(\vec{w}_3 \cdot \vec{x} + b_3)$ 0.2
[197, 184, 136, 214]
$\vec{a}$
[0.3, 0.7, 0.2]

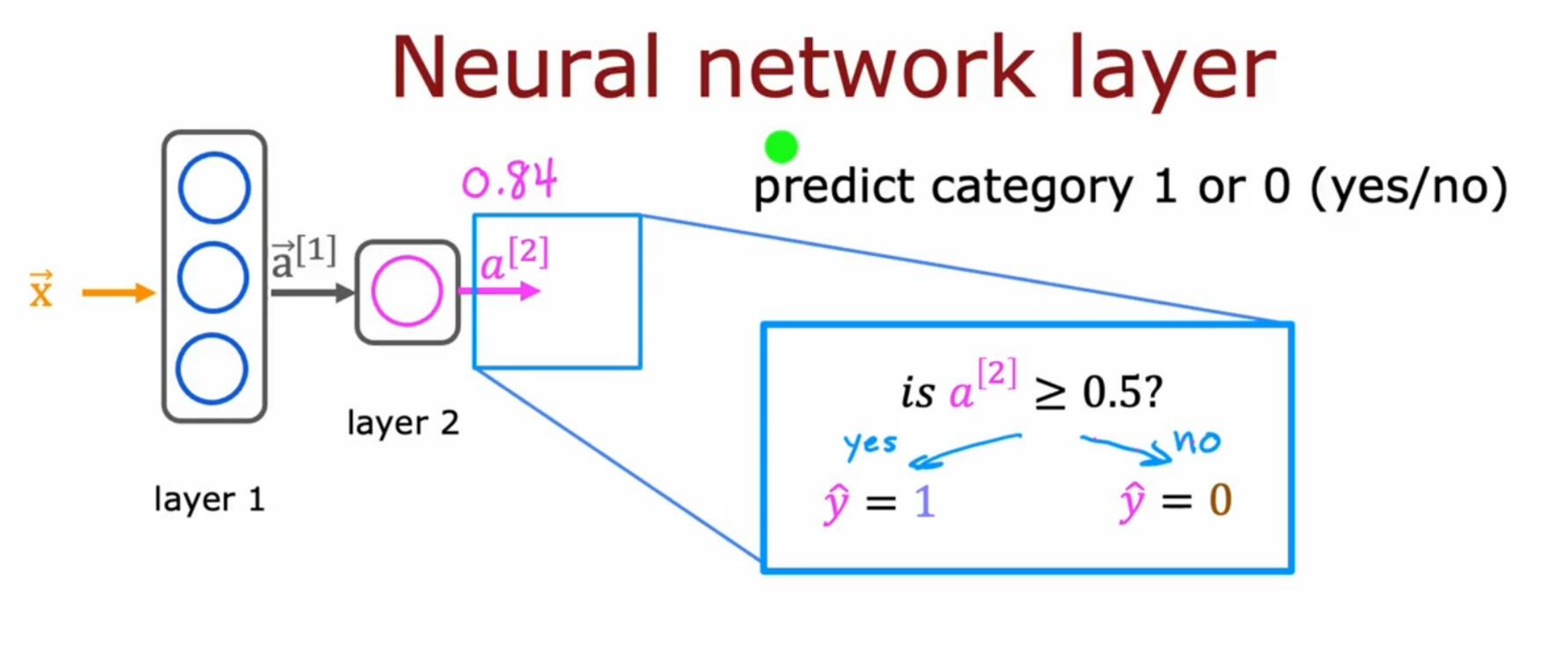
NN Layer


Neural network layer
$g(z) = \frac{1}{1+e^{-(z)}}$
$\vec{x}$
layer 0
$\vec{a}^{[1]}$
layer 2
layer 1
[1] [2]
notation for layer numbering
[197, 184, 136, 214]
$\vec{w}_1^{[1]}, b_1^{[1]}$ $a_1^{[1]} = g(\vec{w}_1^{[1]} \cdot \vec{x} + b_1^{[1]})$ 0.3
z
$\vec{w}_2^{[1]}, b_2^{[1]}$ $a_2^{[1]} = g(\vec{w}_2^{[1]} \cdot \vec{x} + b_2^{[1]})$ 0.7
$\vec{w}_3^{[1]}, b_3^{[1]}$ $a_3^{[1]} = g(\vec{w}_3^{[1]} \cdot \vec{x} + b_3^{[1]})$ 0.2
$\vec{a}^{[1]}$
vector of activation values from layer 1
[0.3, 0.7, 0.2]

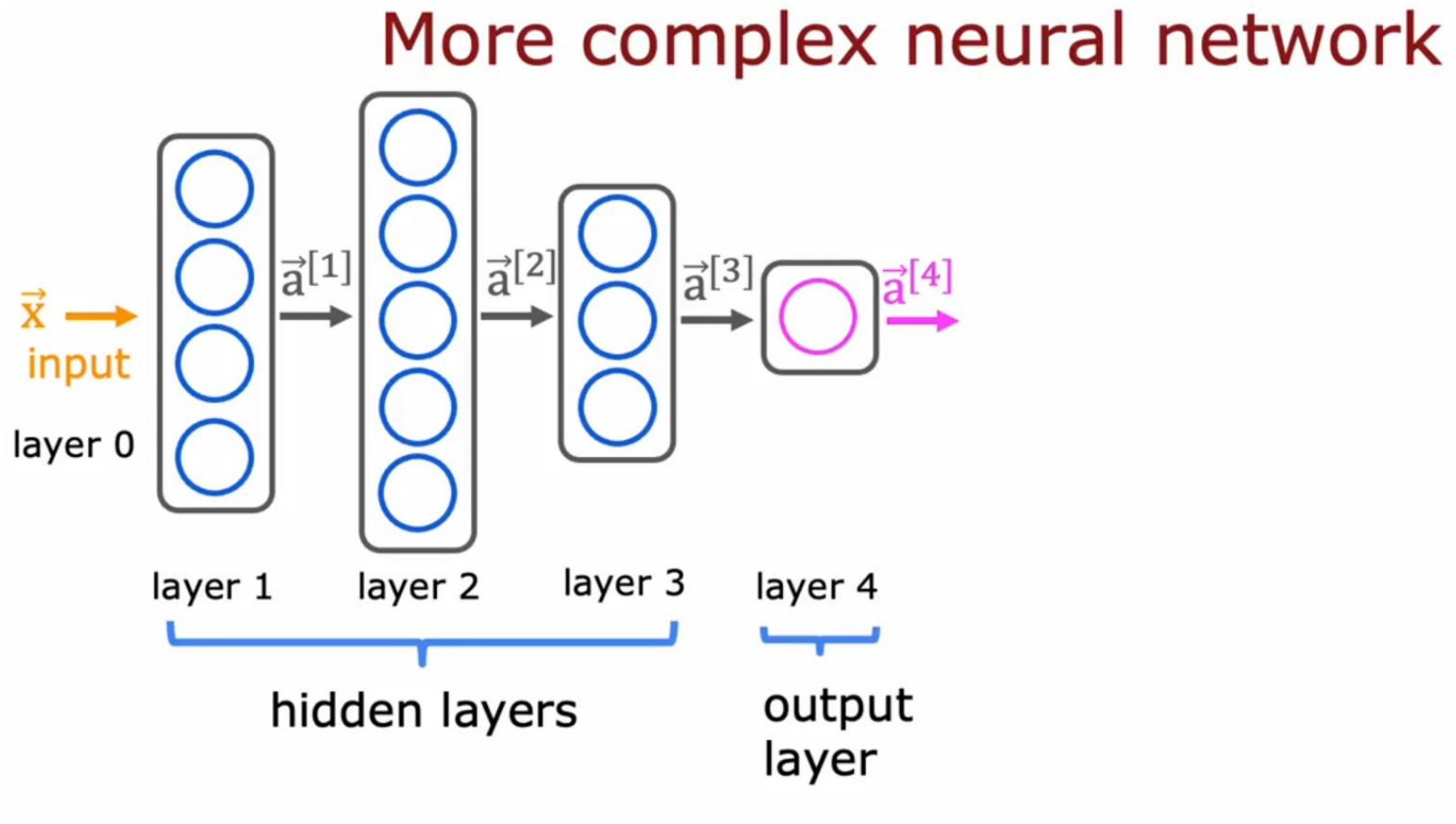
NN Layer


Neural network layer
$g(z) = \frac{1}{1+e^{-(z)}}$
$\vec{x}$
$\vec{a}^{[1]}$
$a^{[2]}$
layer 2
layer 1
0.3
0.7
0.2
$\vec{w}_1$, b_1
$a_1 = g(\vec{w}_1 \cdot \vec{a}^{[1]} + b_1)$
z
0.84
a scalar value

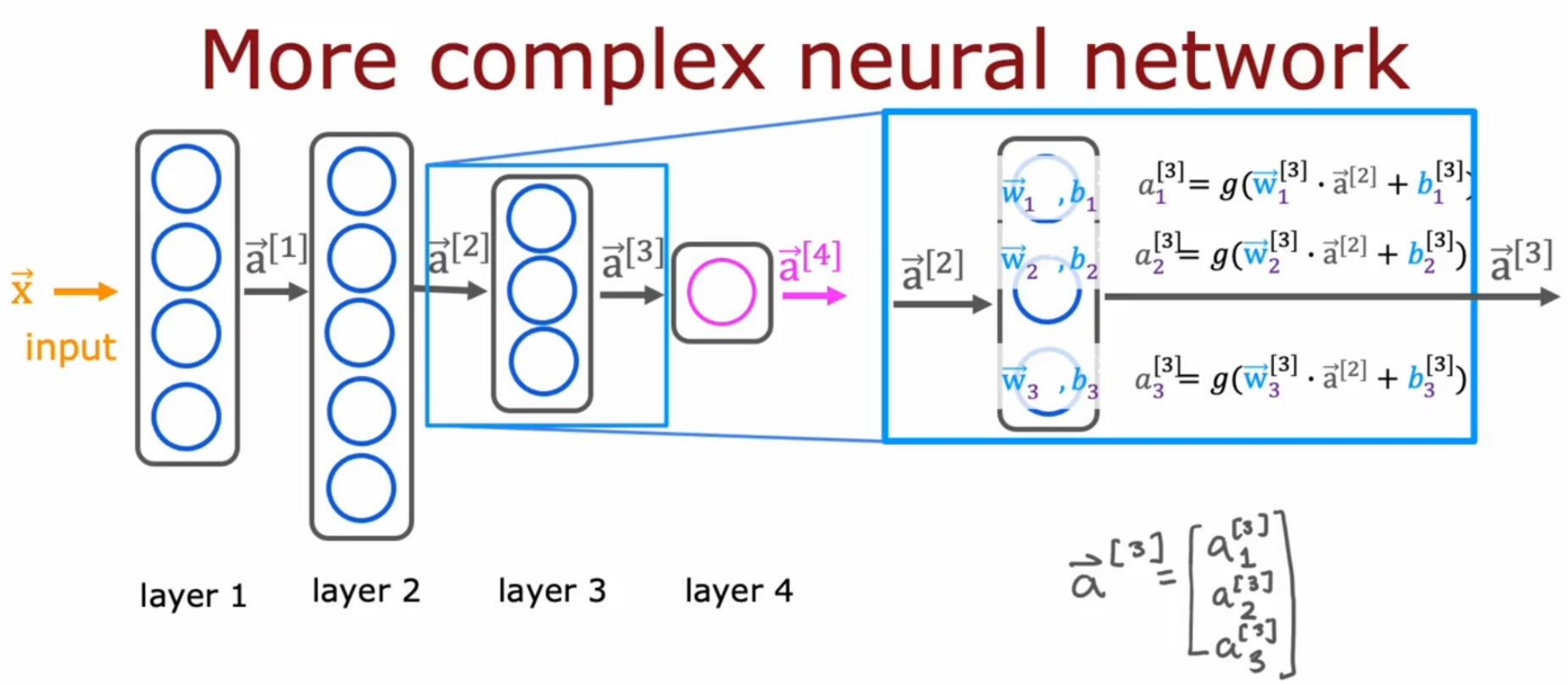
NN Layer


Neural network layer
predict category 1 or 0 (yes/no)
0.84
$\vec{x}$
$\vec{a}^{[1]}$
$a^{[2]}$
layer 1
layer 2
is $a^{[2]} \geq 0.5$?
yes
no
$\hat{y} = 1$
$\hat{y} = 0$

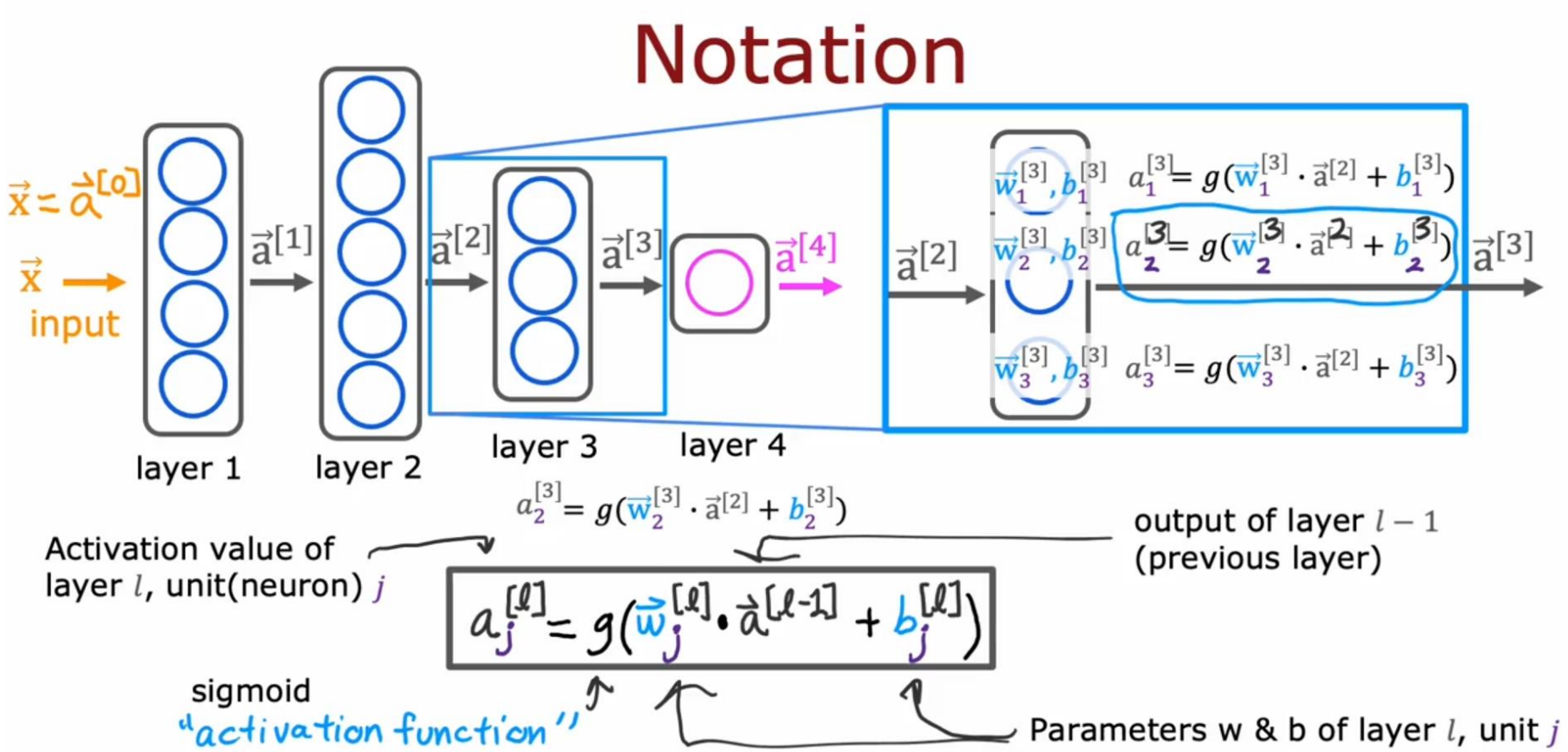
More Complex NN



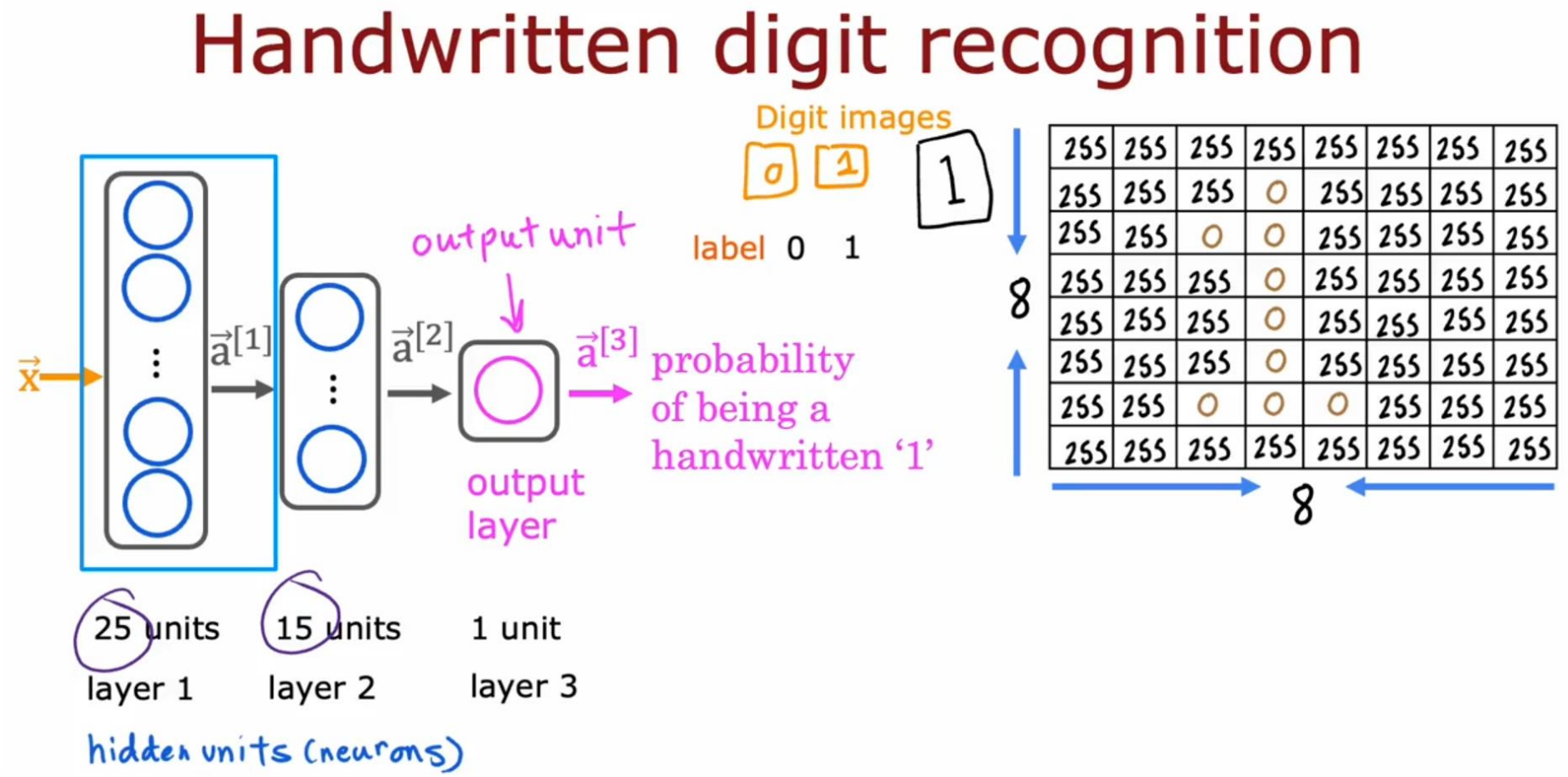
More Complex NN



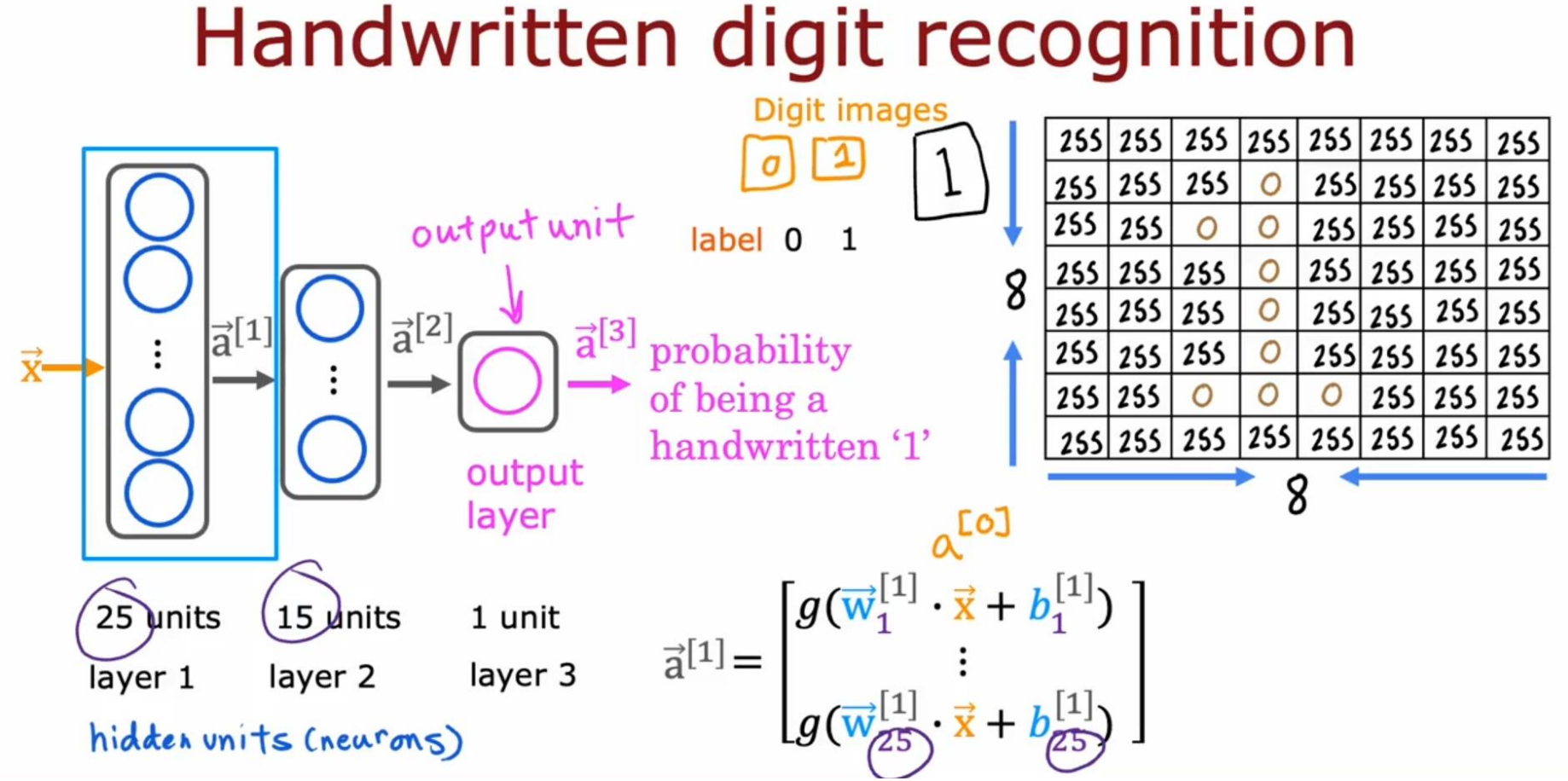
More Complex NN



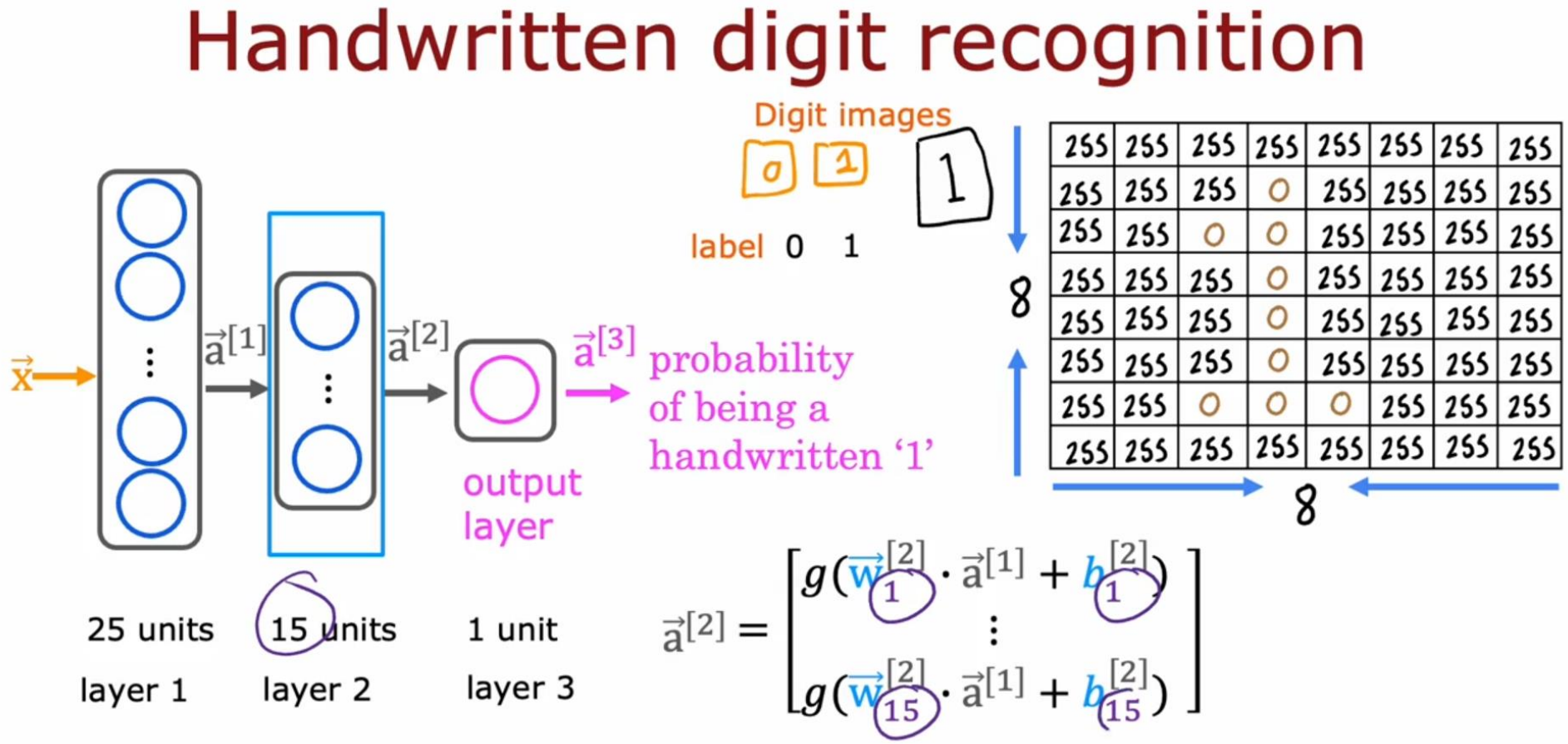
Inference: Making Predictions



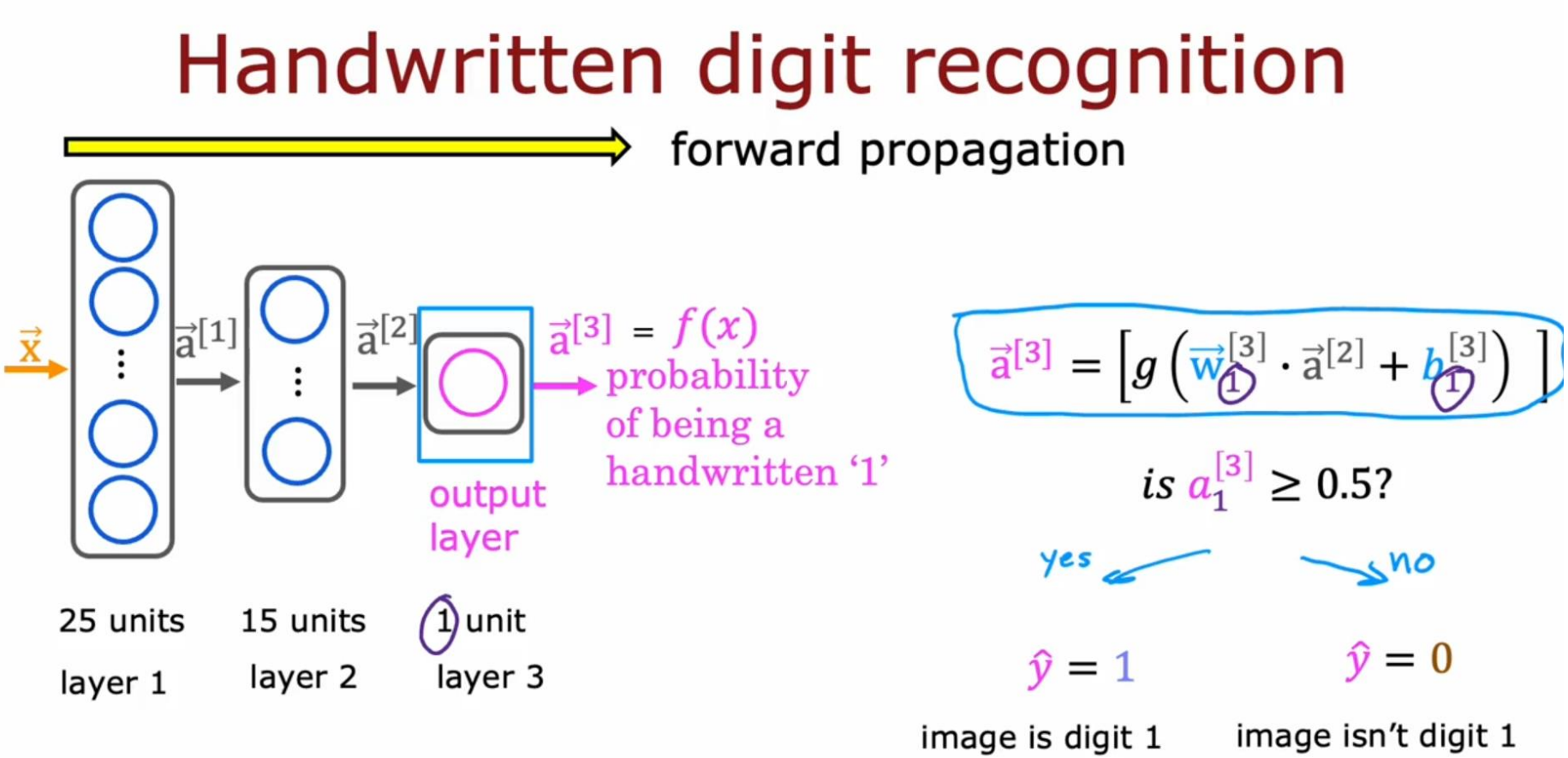
Inference: Making Predictions



Inference: Making Predictions

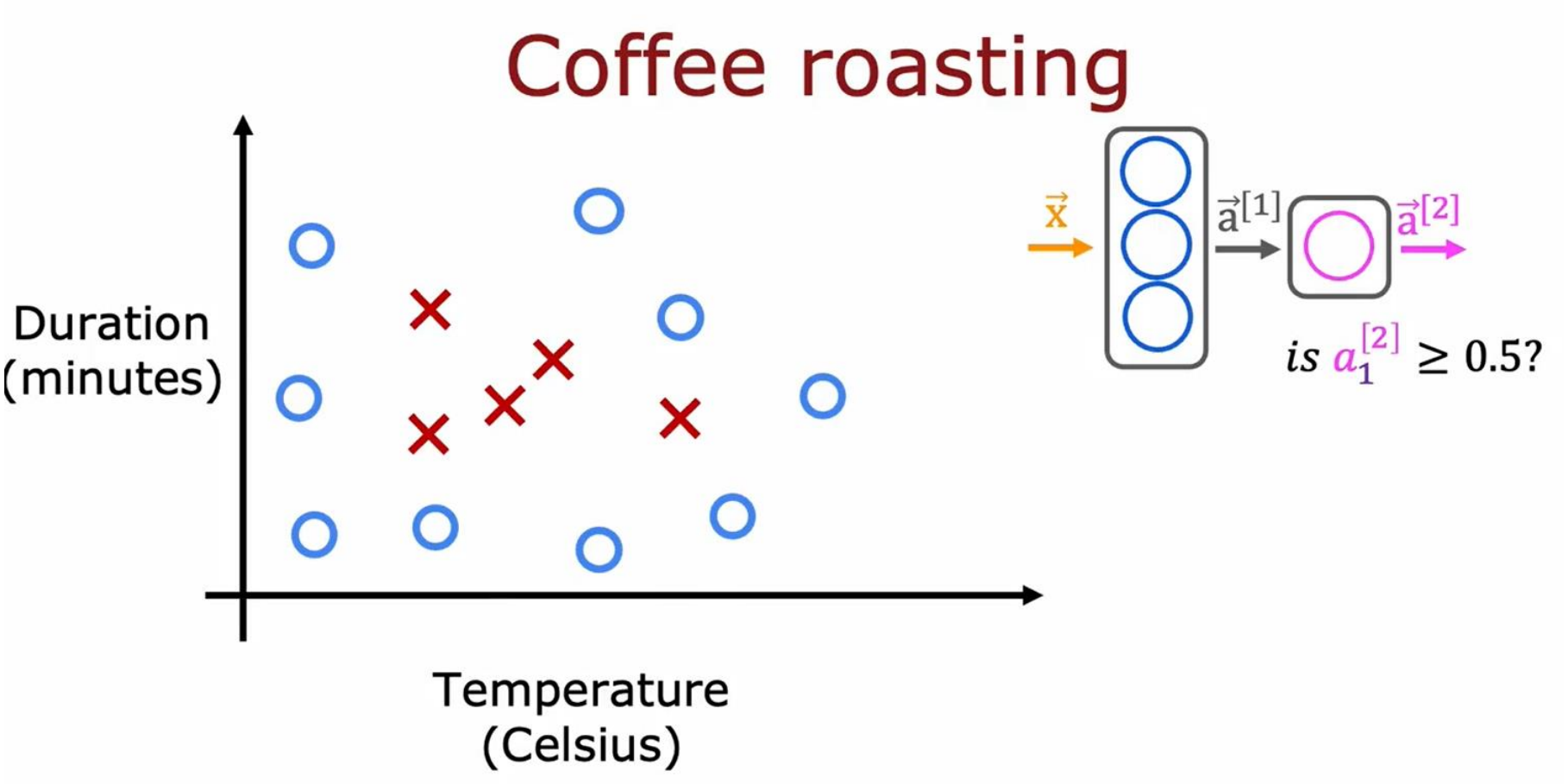


Inference: Making Predictions

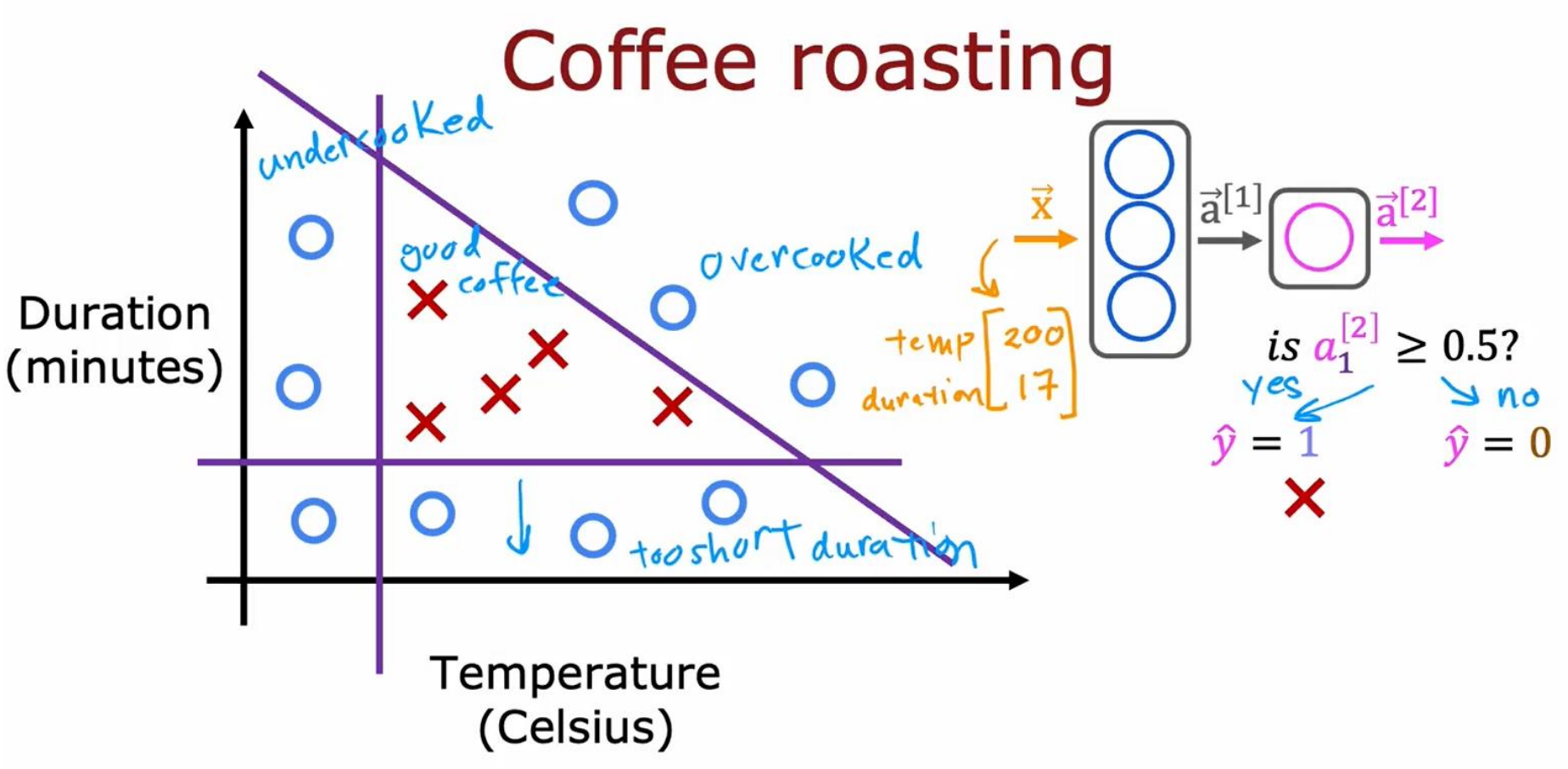


3. TensorFlow Implementation

Inference in Code

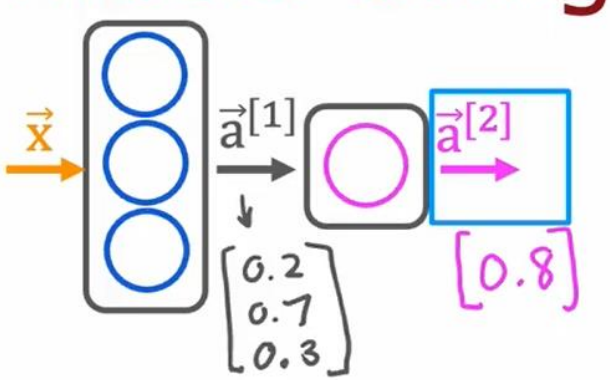


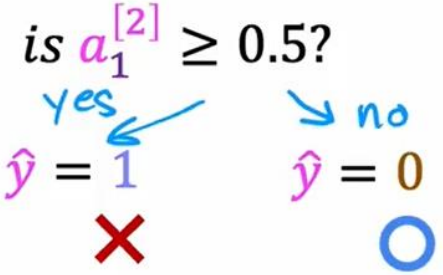
Inference in Code



Inference in Code

Build the model using TensorFlow





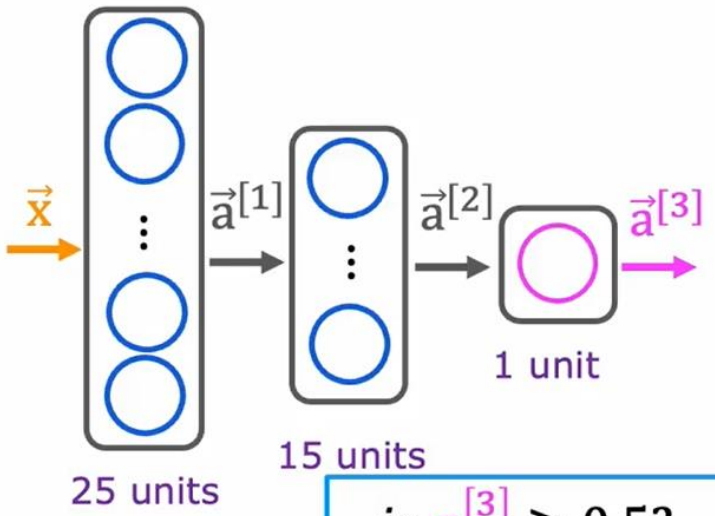

```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation='sigmoid')
a1 = layer_1(x)
```

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

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

Inference in Code

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)
```

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

$\hat{y} = 1$ ✗ $\hat{y} = 0$ ○

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

Data in TensorFlow

Feature vectors

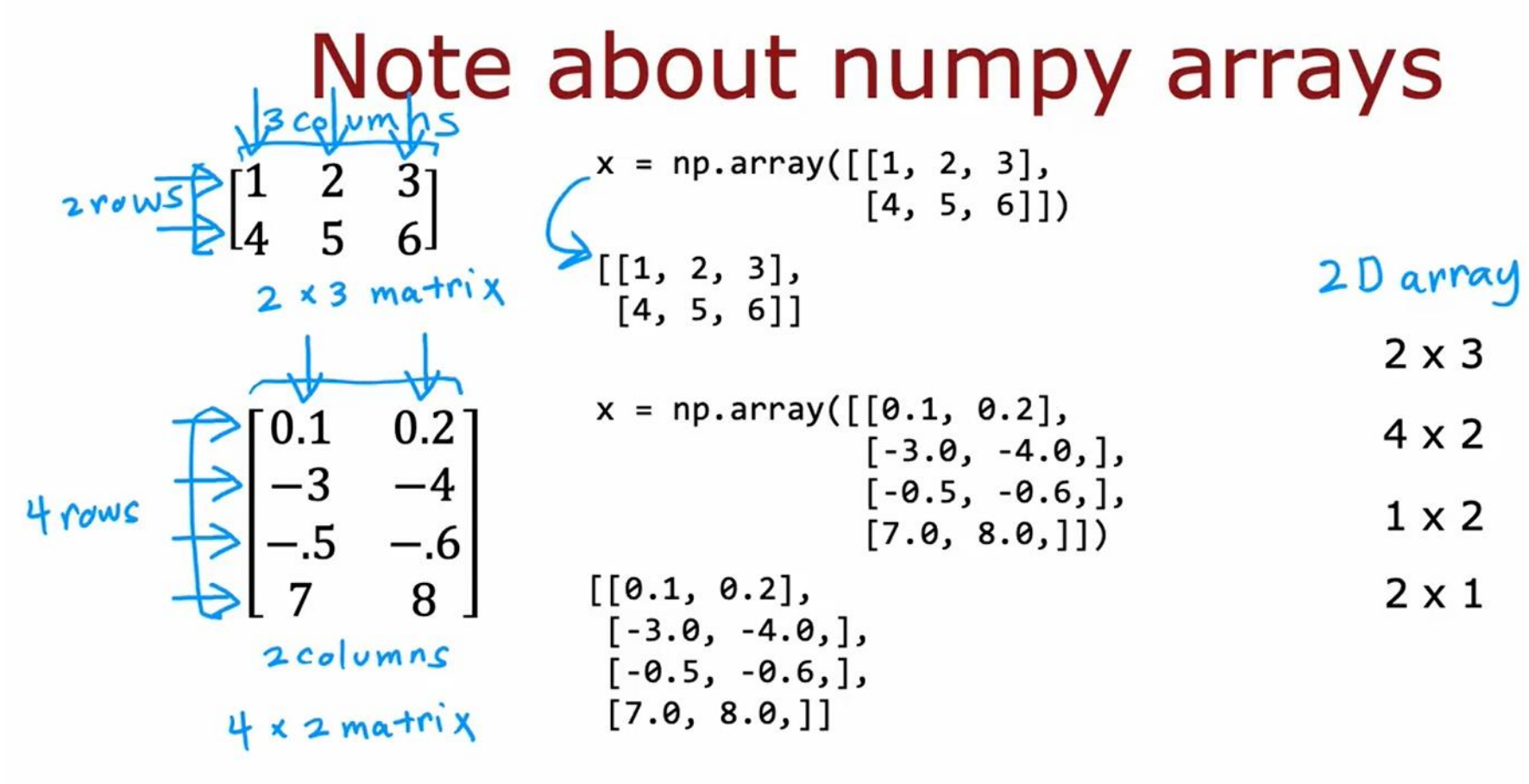
temperature (Celsius)	duration (minutes)	Good coffee? (1/0)
200.0	17.0	1
425.0	18.5	0
...	...	...

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

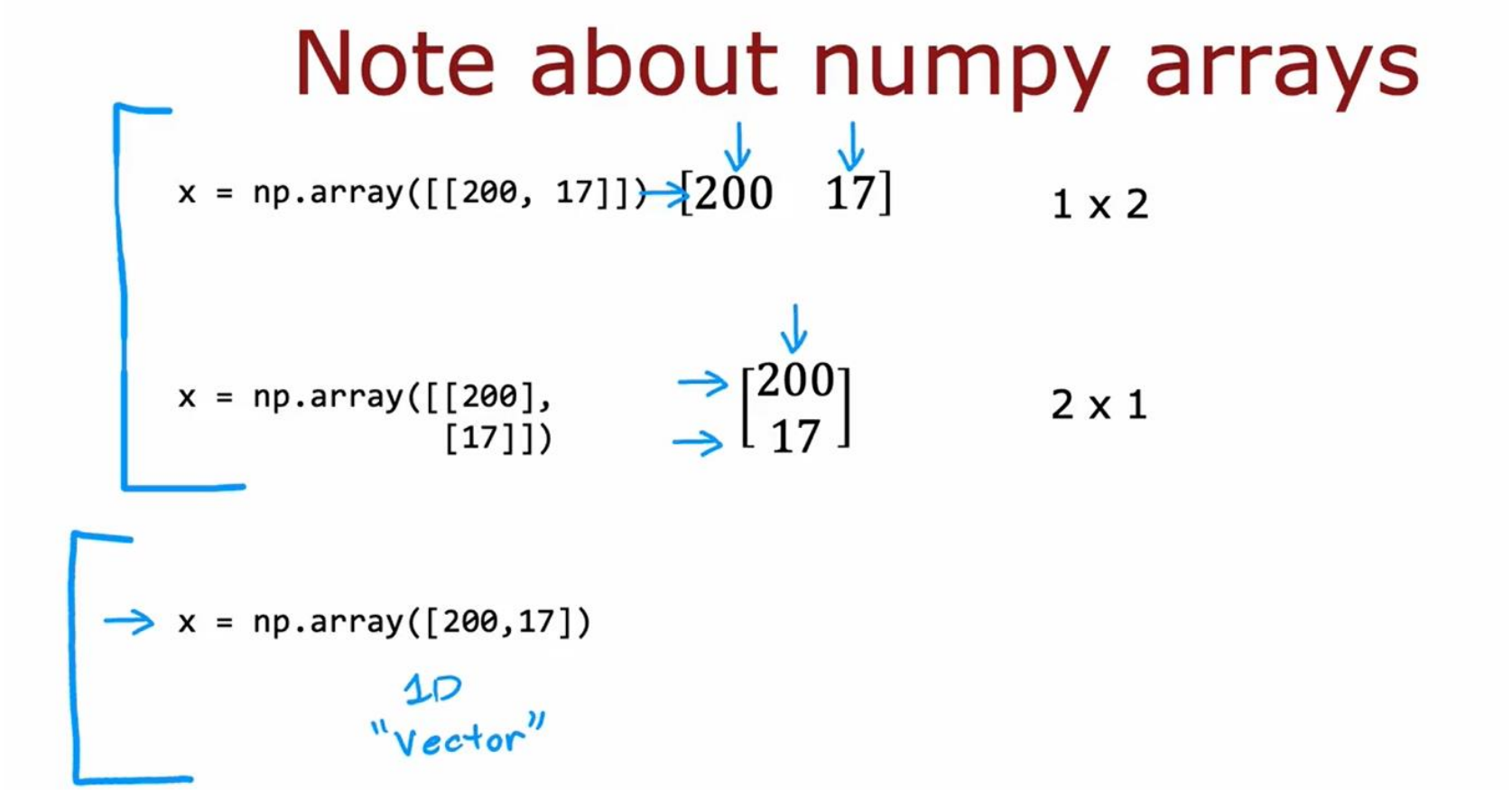
```
[[200.0, 17.0]]
```

Why?

Data in TensorFlow



Data in TensorFlow



Data in TensorFlow

Feature vectors

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

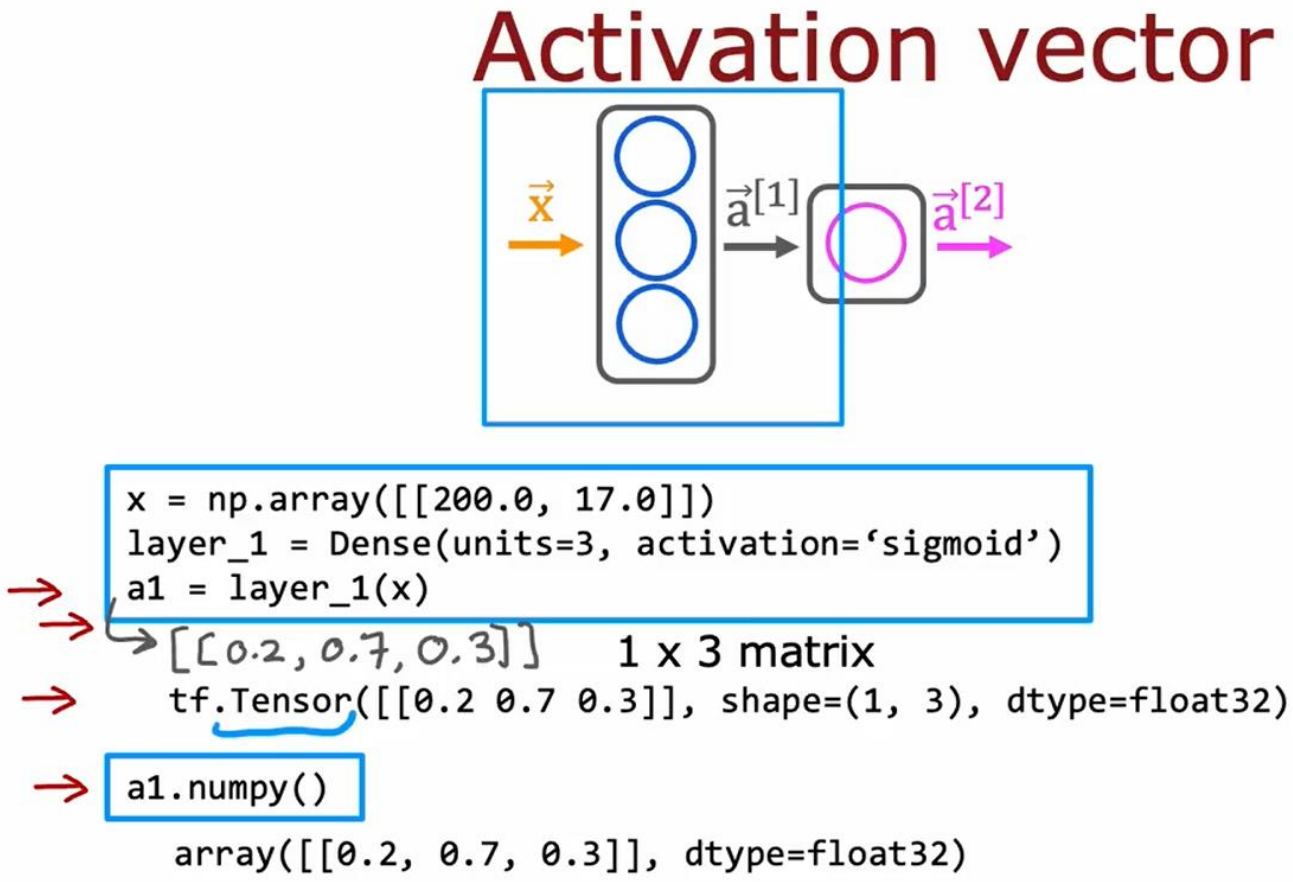
```
x = np.array([[200.0, 17.0]])
```

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

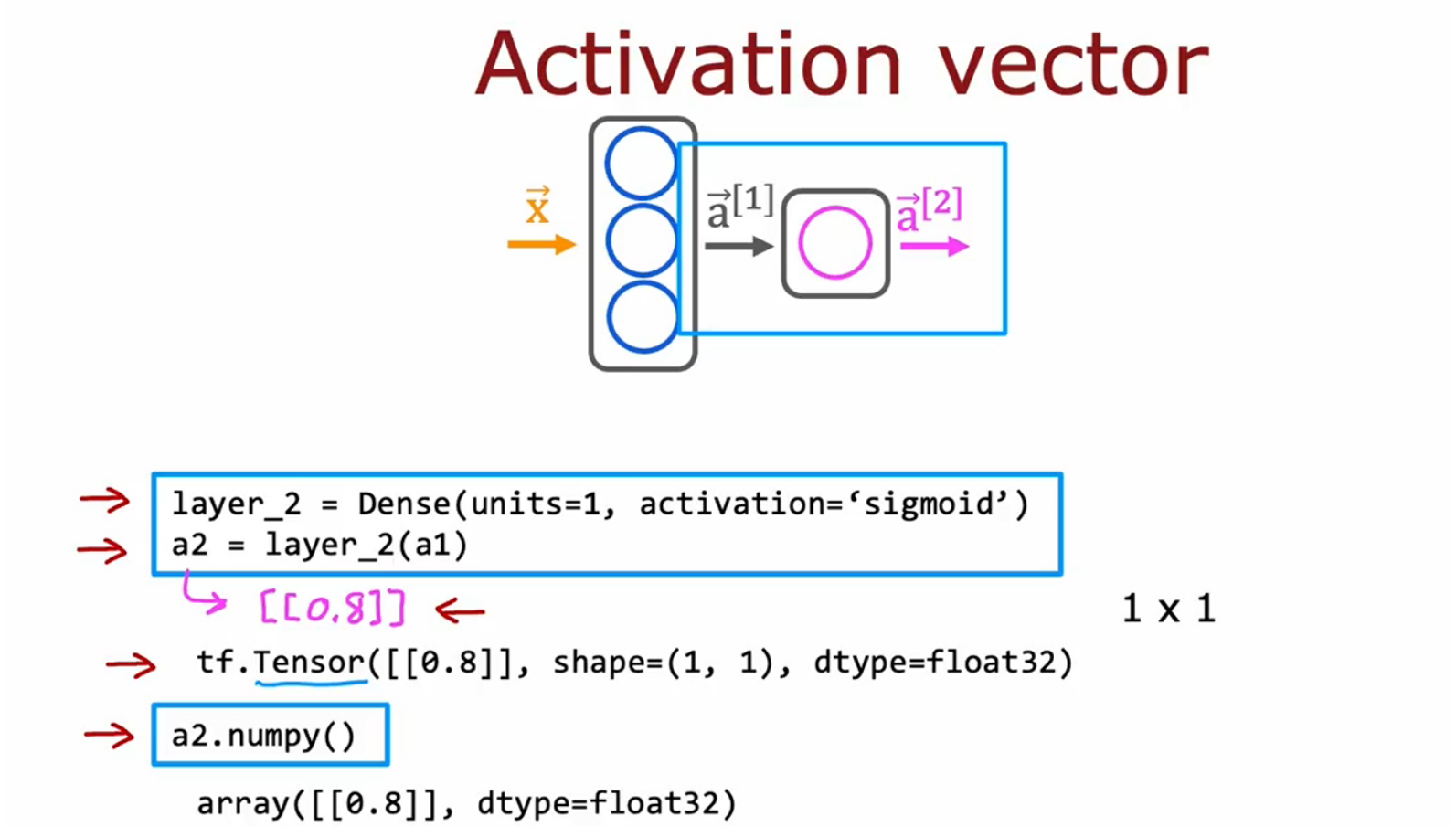
1 x 2

[200.0 17.0]

Data in TensorFlow

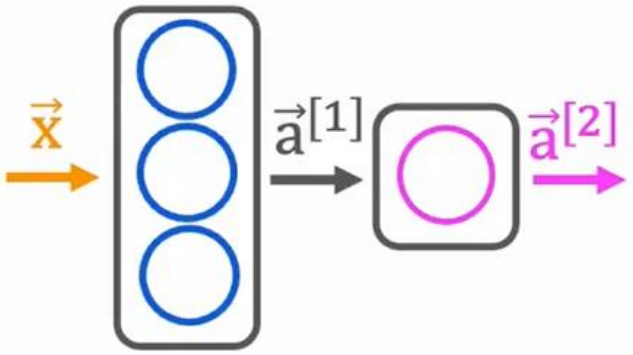


Data in TensorFlow



Building a Neural Network

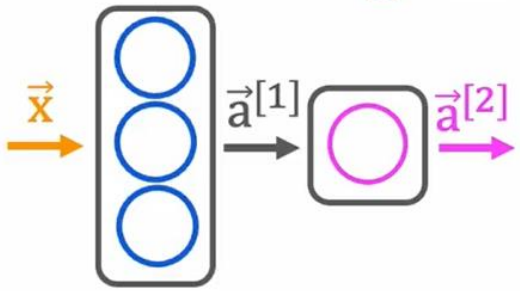
What you saw earlier




```
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

Building a neural network architecture




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

layer

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

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

4 x 2

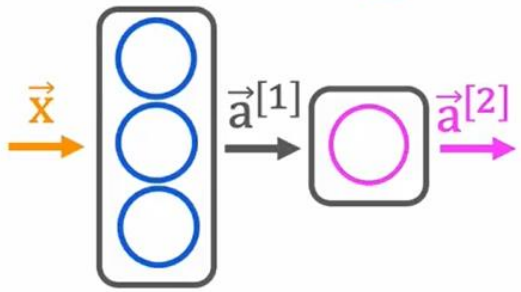
targets

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

more about this next week!

Building a Neural Network

Building a neural network architecture




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

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

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

4 x 2

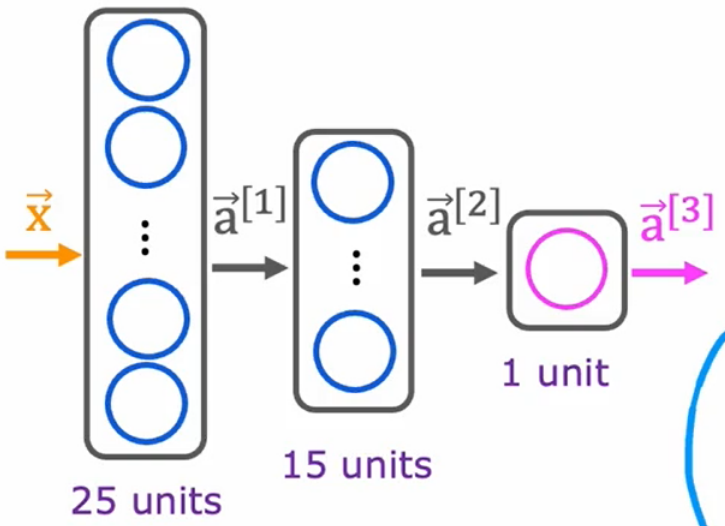
targets

```
y = np.array([1,0,0,1])

model.compile(...)
model.fit(x,y)

model.predict(x_new)
```

← more about this next week!

Building a Neural Network

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)
```