Copyright Notice

These slides are distributed under the Creative Commons License.

DeepLearning.AI makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite DeepLearning.AI as the source of the slides.

For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode

Neural Networks

Biological neuron

inputs outputs

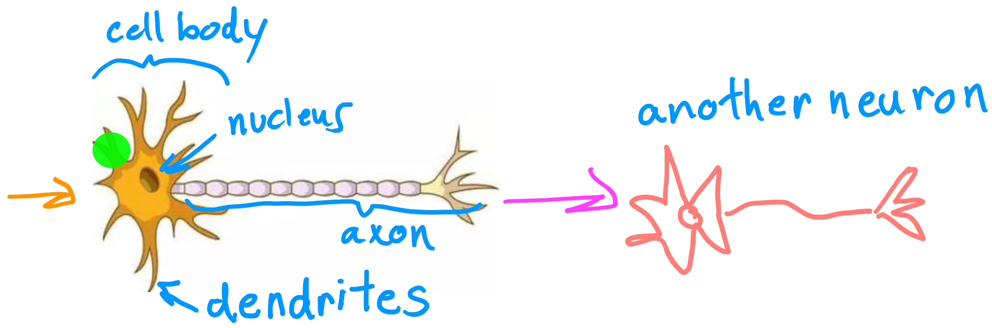
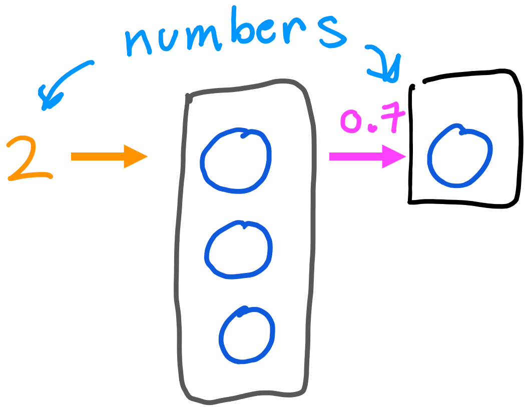


image source: https://biologydictionary.net/sensory-neuron/

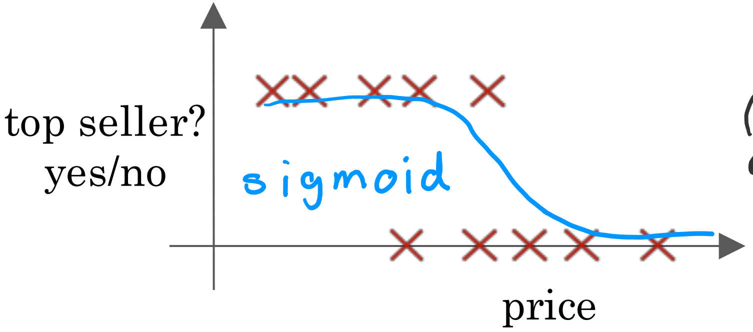
Simplified mathematical model of a neuron

inputs outputs



Andrew Ng

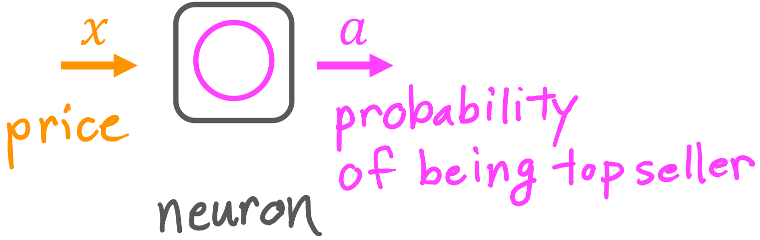
Demand Prediction



$x = price$ input

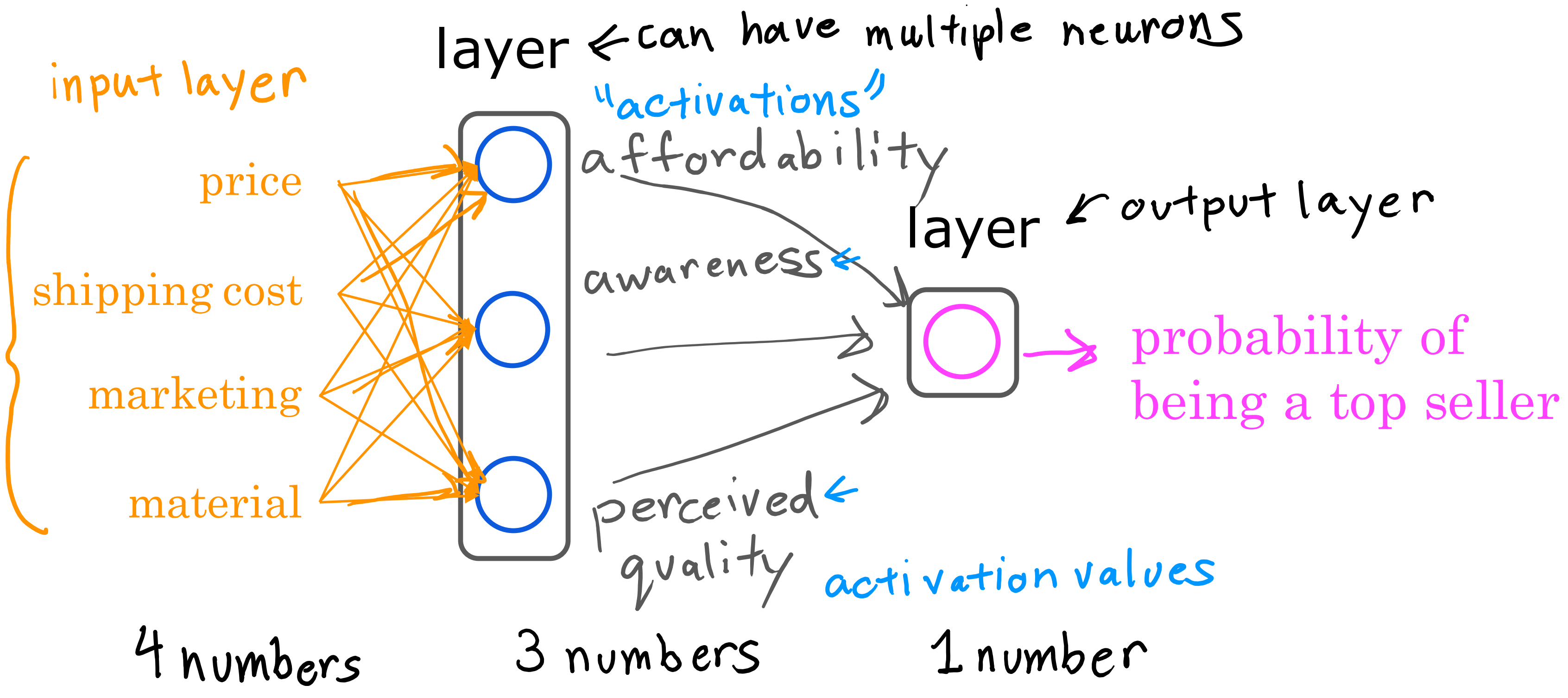
$a = f(x) = \dfrac{1}{1 + e^{-(wx+b)}}$ output

activation



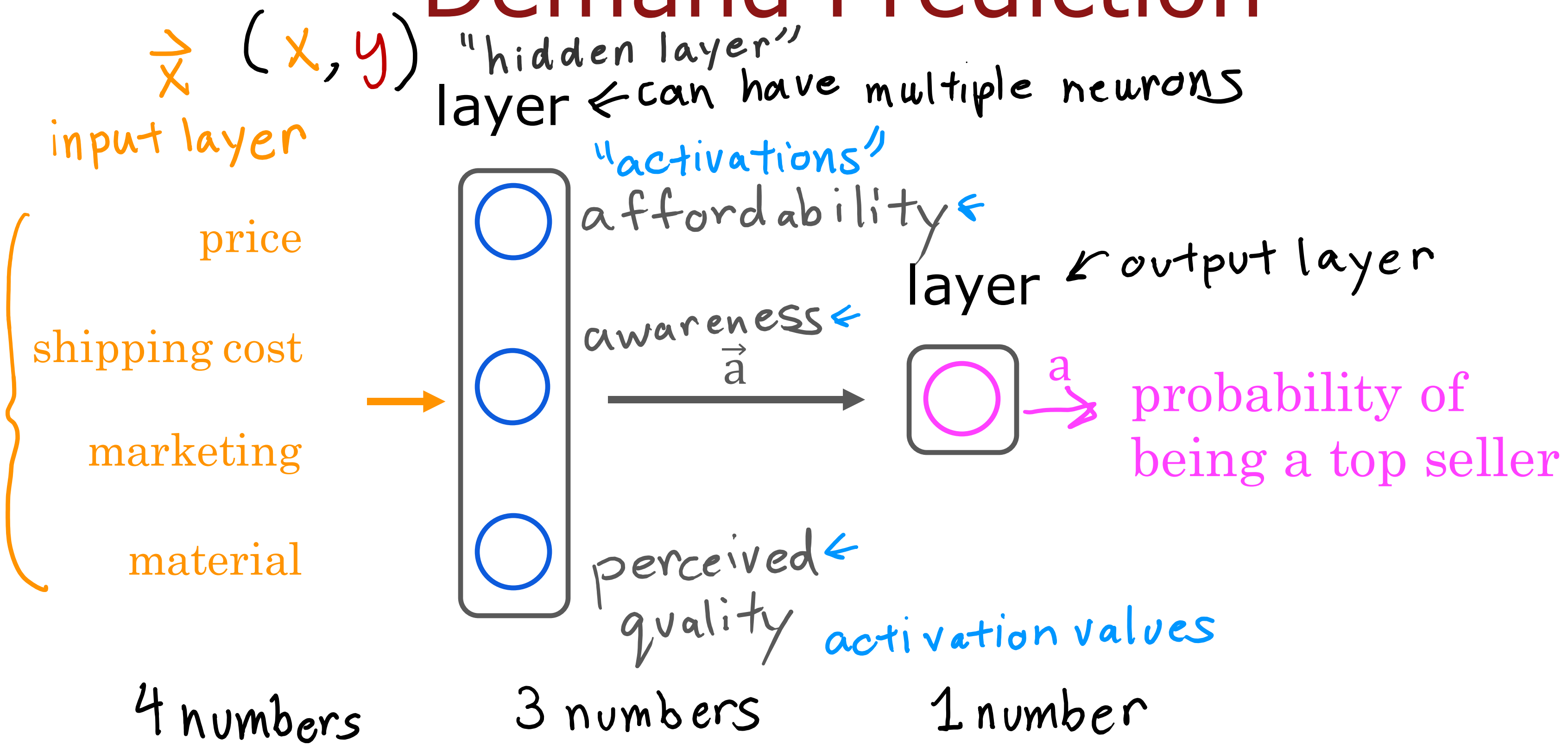
Andrew Ng

Demand Prediction



Andrew Ng

Demand Prediction



Andrew Ng

Demand Prediction

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

input layer

"hidden layer"

layer ← can have multiple neurons

"activations"

feature engineering

$x_1 x_2$

price

shipping cost

marketing

material

affordability ←

awareness ←

$\vec{a}$

perceived quality ←

layer ← output layer

a

probability of being a top seller

activation values

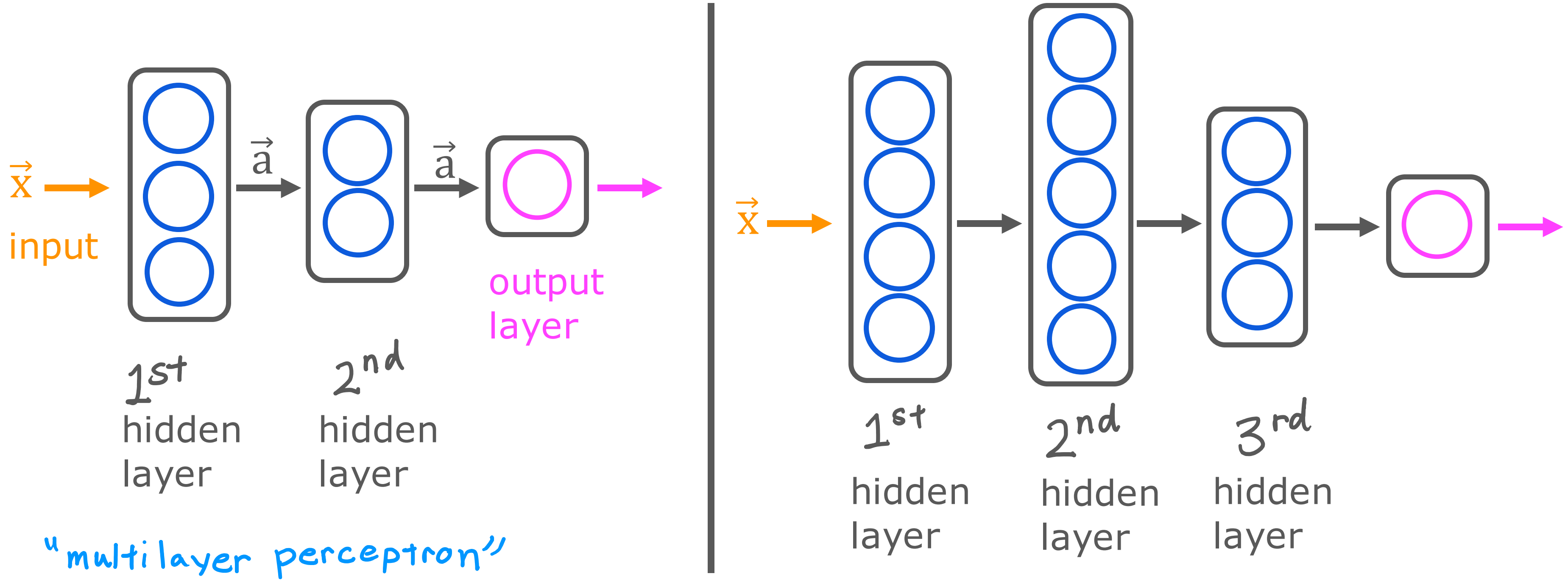
4 numbers

3 numbers

1 number

Andrew Ng

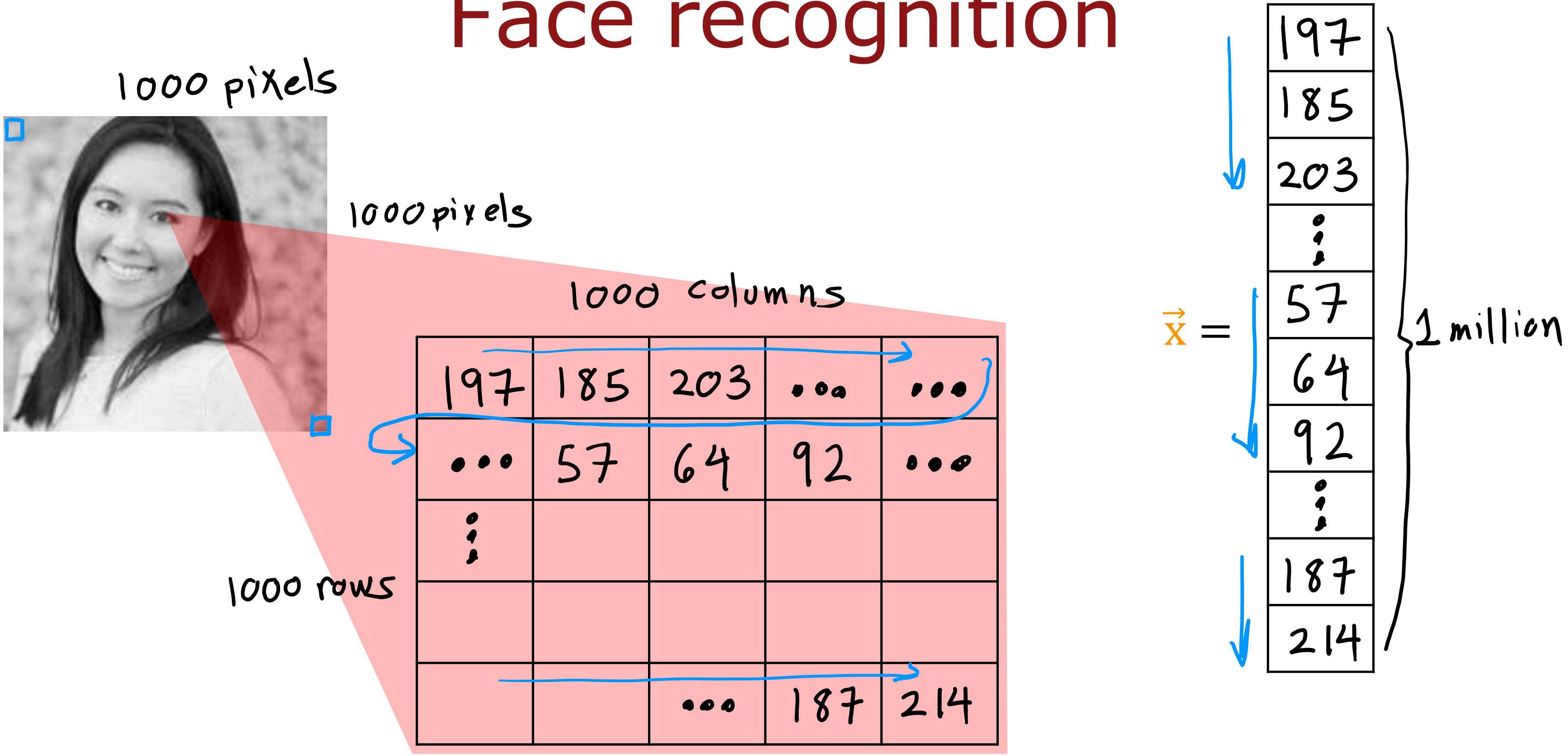
Multiple hidden layers



neural network architecture

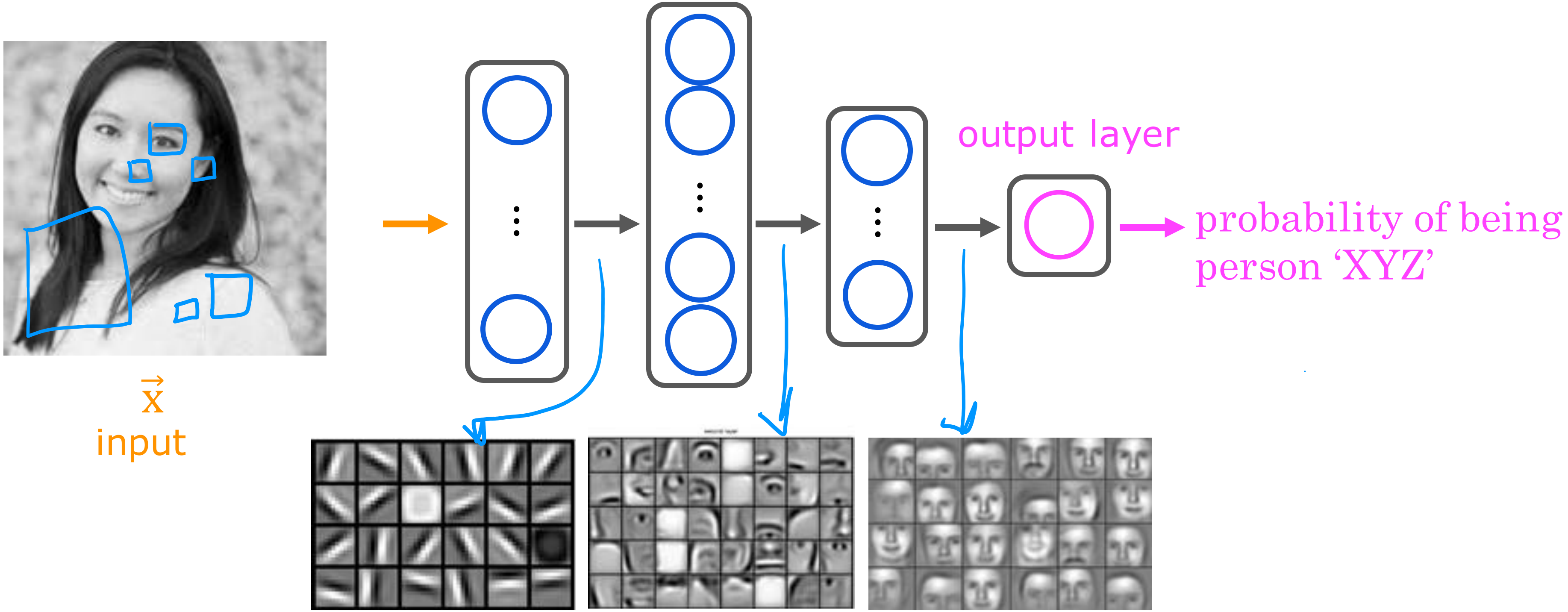
Andrew Ng

Face recognition



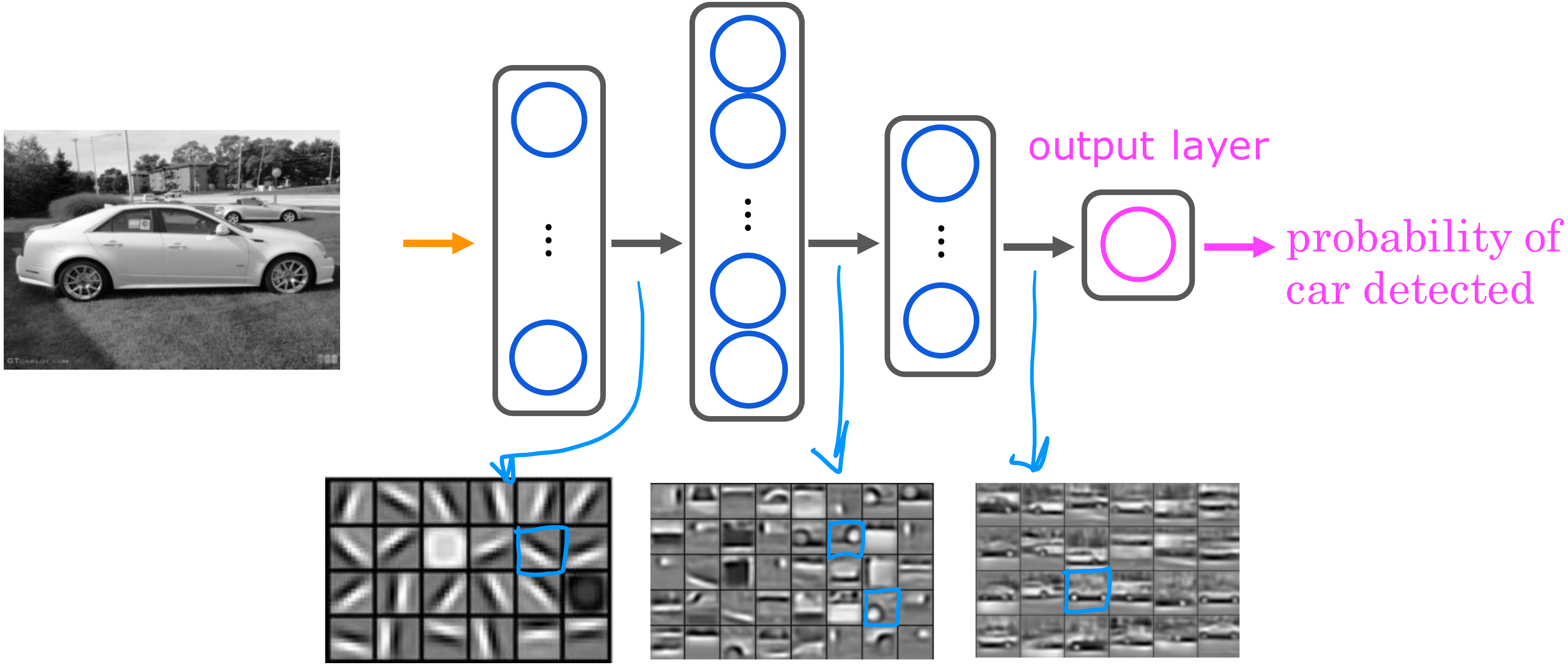
Andrew Ng

Face recognition



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

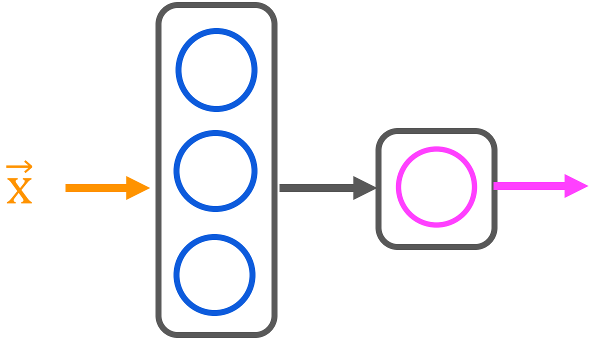
Car classification



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

Andrew Ng

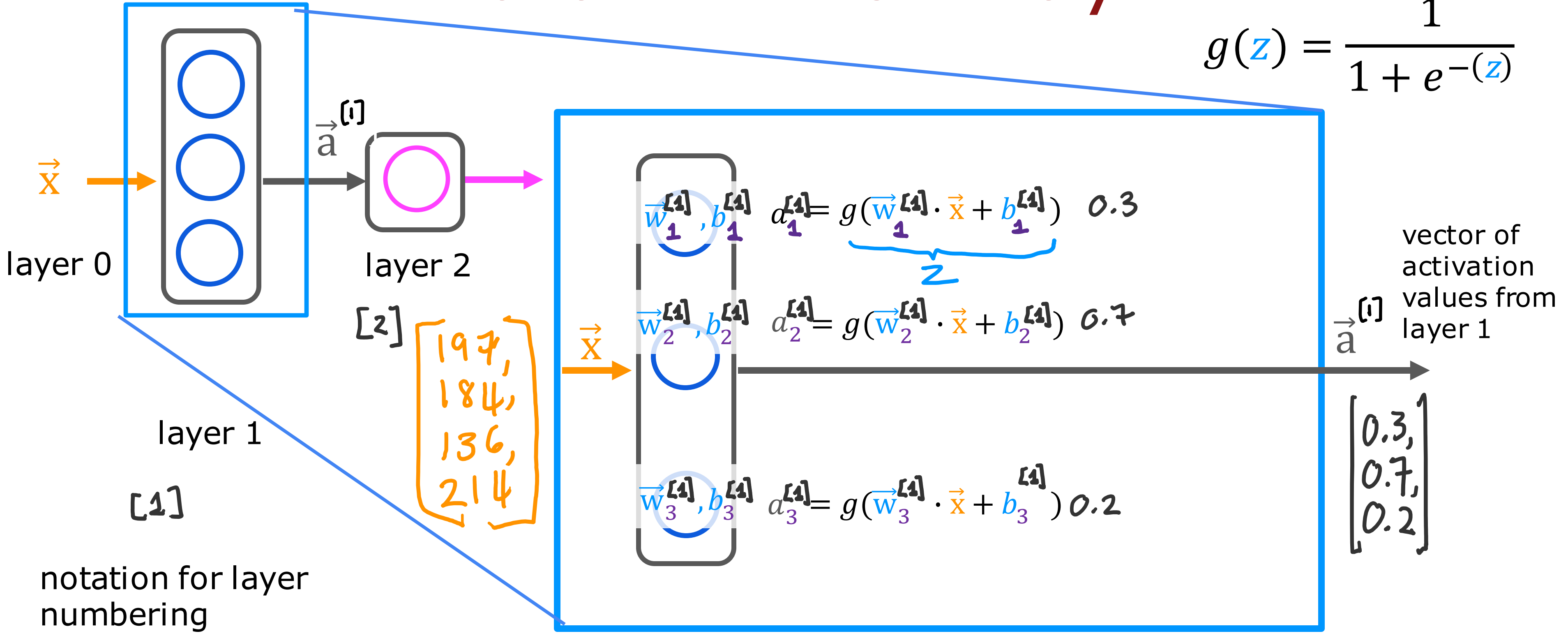
Neural network layer



Andrew Ng

Neural network layer

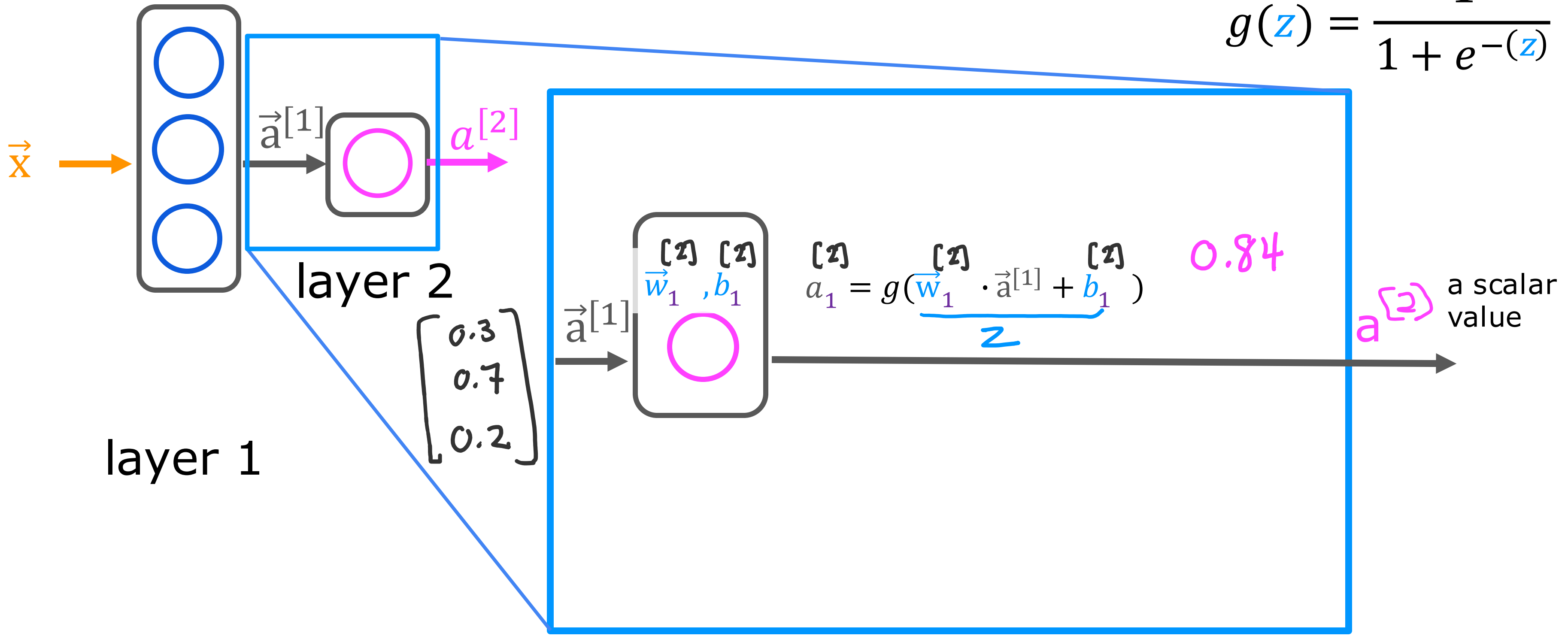
$$g(z) = \frac{1}{1 + e^{-(z)}}$$



Andrew Ng

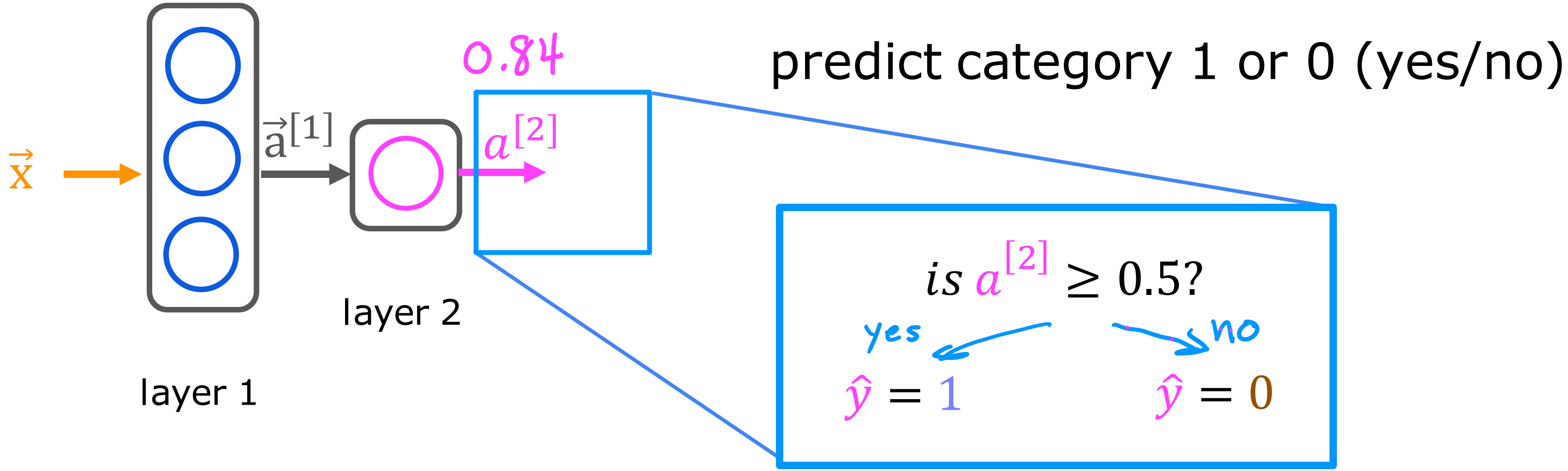
Neural network layer

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



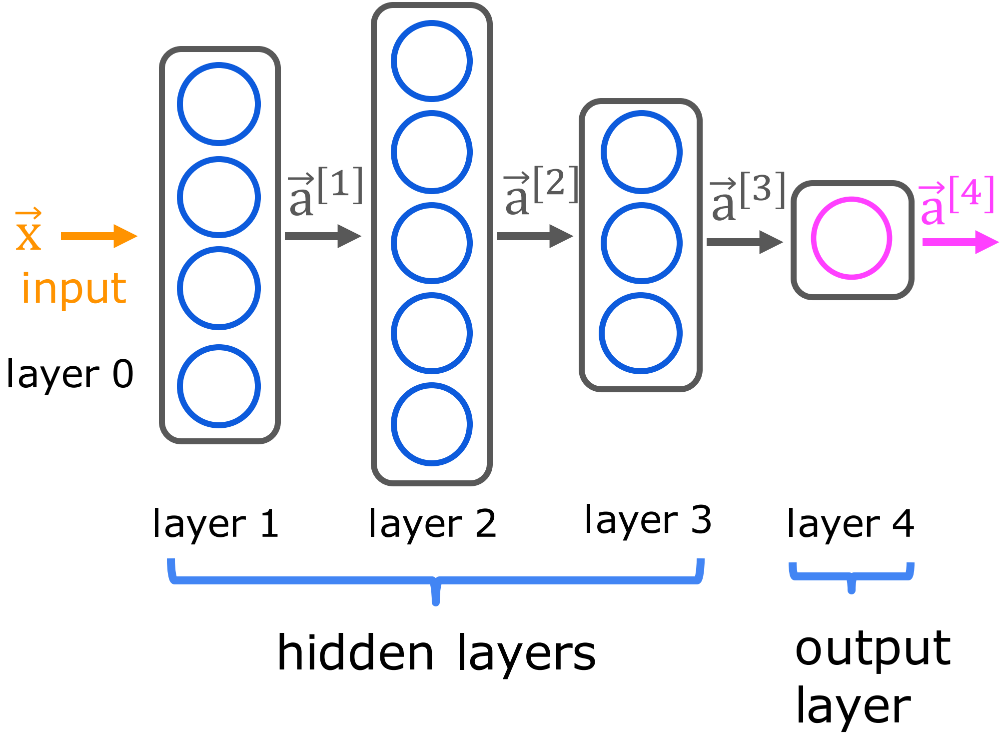
Andrew Ng

Neural network layer



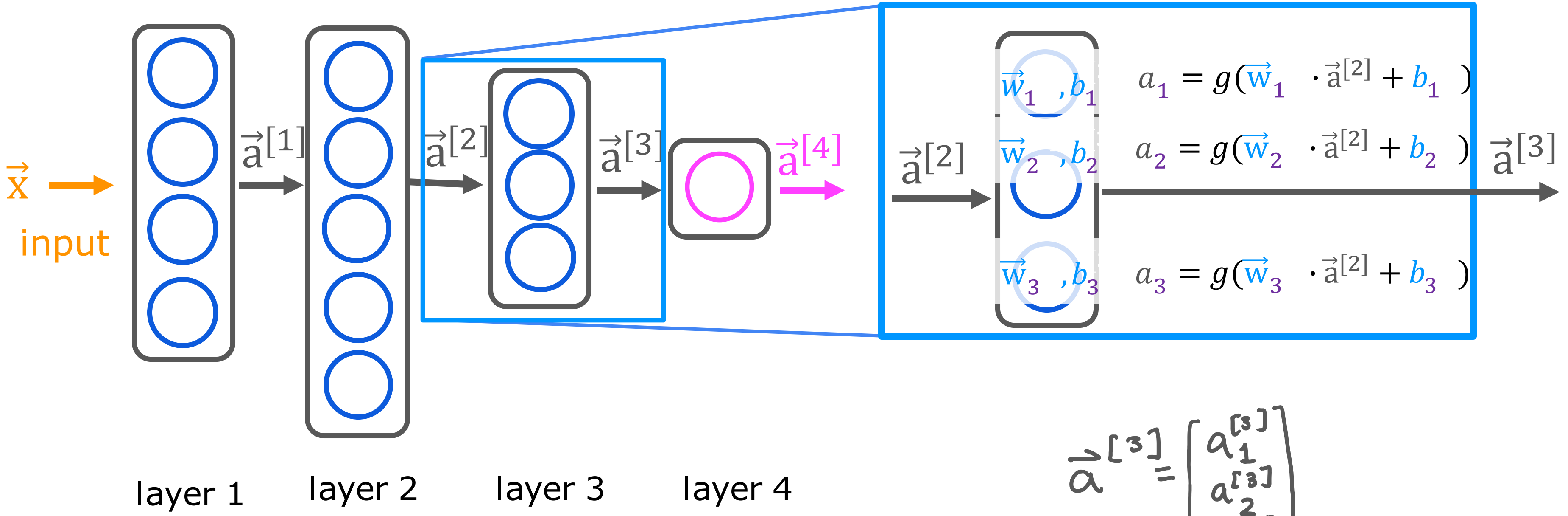
Andrew Ng

More complex neural network



Andrew Ng

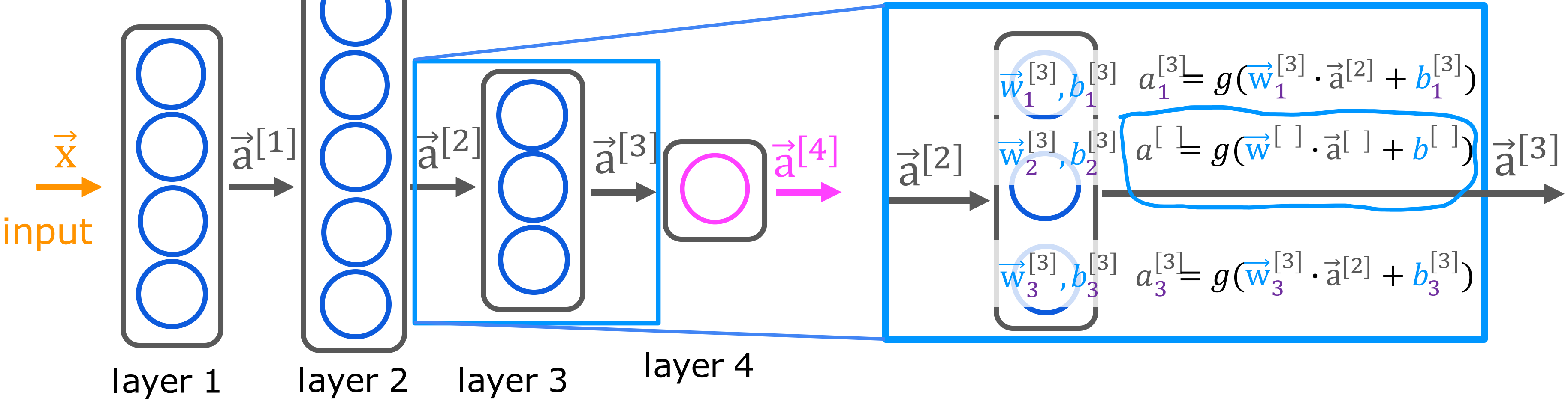
More complex neural network



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

Andrew Ng

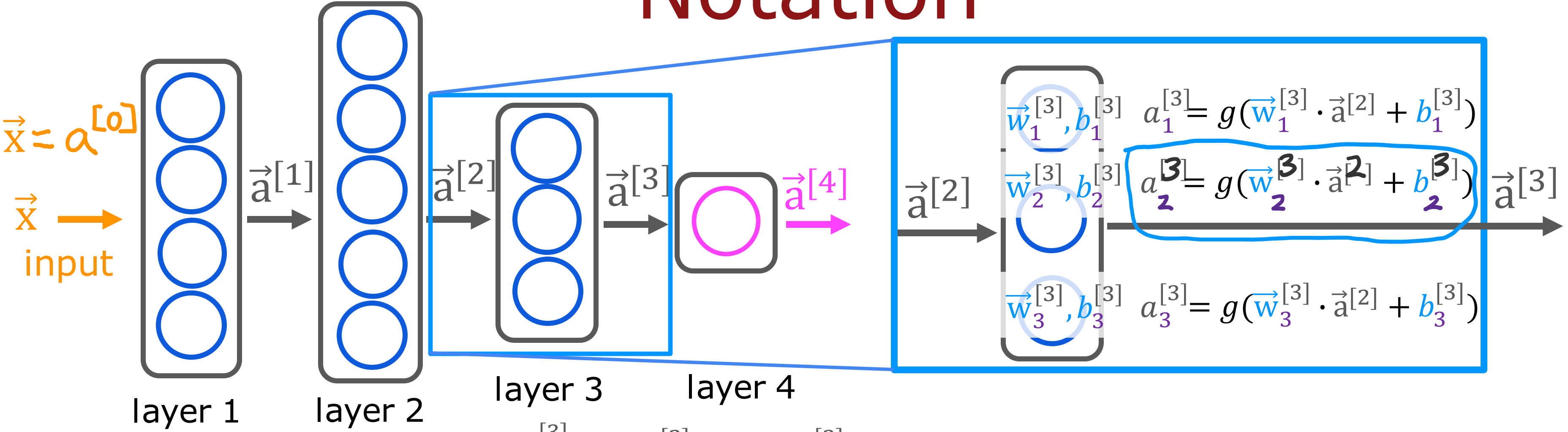
Notation



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

Andrew Ng

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)

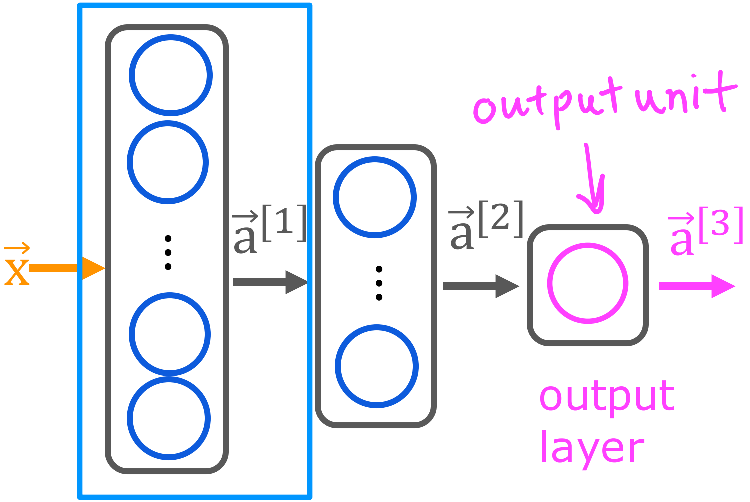
$$\boxed{a_j^{[l]} = g(\vec{w}_j^{[l]} \cdot \vec{a}^{[l-1]} + b_j^{[l]})}$$

sigmoid
"activation function"

Parameters w & b of layer l, unit j

Andrew Ng

Handwritten digit recognition



25 units	15 units	1 unit
layer 1	layer 2	layer 3

hidden units (neurons)

Digit images

label 0 1

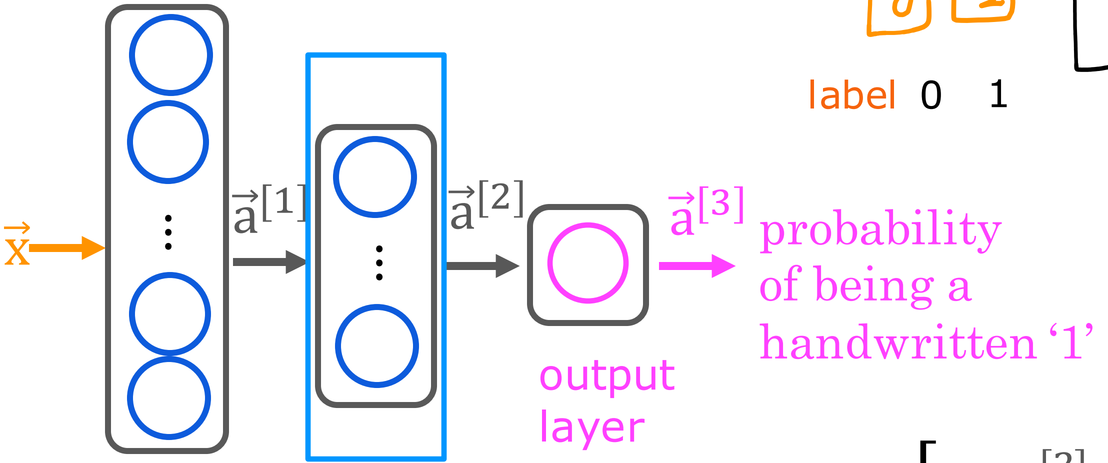
8 × 8

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

$a^{[0]}$

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

Handwritten digit recognition



25 units	15 units	1 unit
layer 1	layer 2	layer 3

Digit images

label 0 1

8 × 8

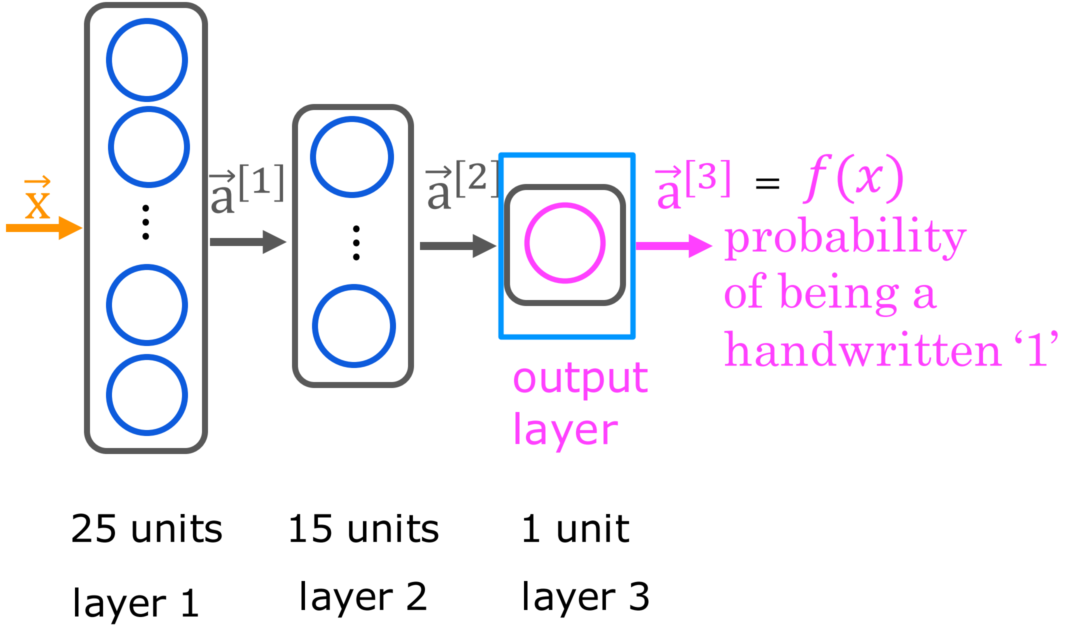
255	255	255	255	255	255	255	255
255	255	255	0	255	255	255	255
255	255	0	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	0	0	0	255	255	255
255	255	255	255	255	255	255	255

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

Andrew Ng

Handwritten digit recognition

forward propagation



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

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

yes no

$\hat{y} = 1$ — image is digit 1

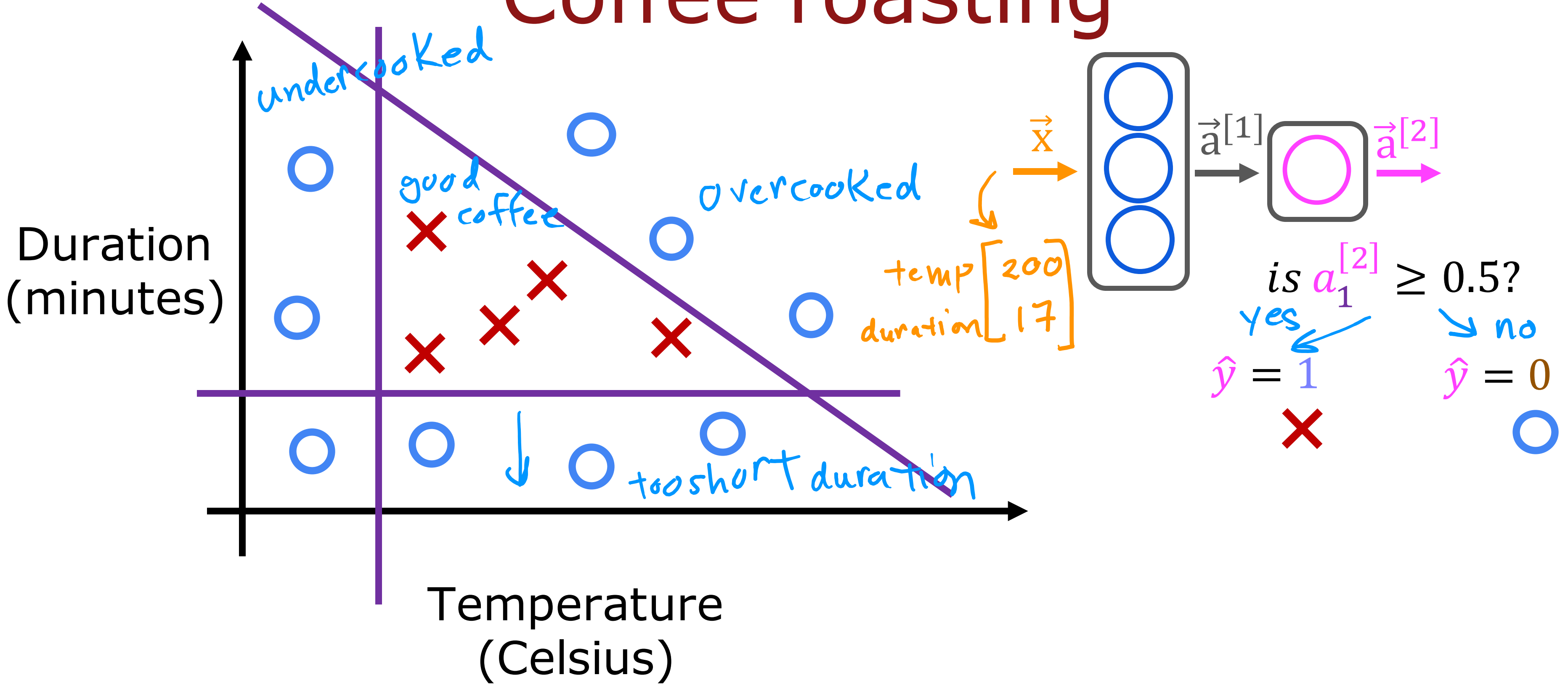
$\hat{y} = 0$ — image isn't digit 1

Andrew Ng

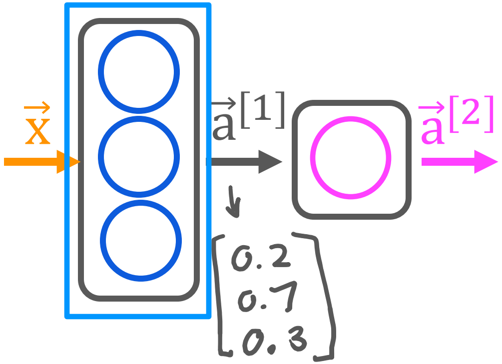
TensorFlow implementation

Inference in Code

Coffee roasting



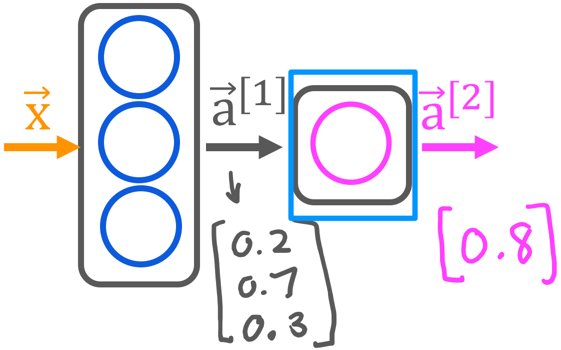
Andrew Ng




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

Andrew Ng

Build the model using TensorFlow

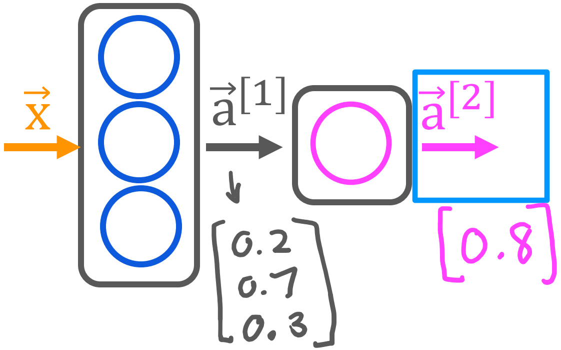



```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation='sigmoid')
a1 = layer_1(x)
```

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

Andrew Ng

Build the model using TensorFlow



$$\text{is } a_1^{[2]} \geq 0.5?$$

yes $\qquad\qquad$ no

$\hat{y} = 1 \qquad\qquad \hat{y} = 0$

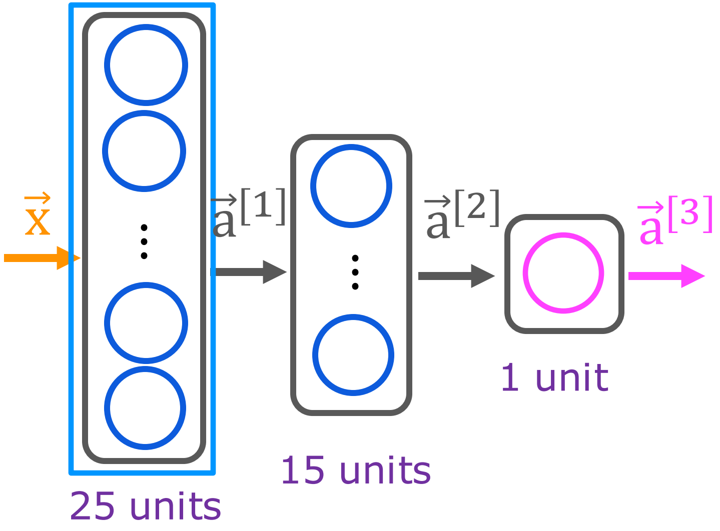
```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation='sigmoid')
a1 = layer_1(x)
```

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

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

Andrew Ng

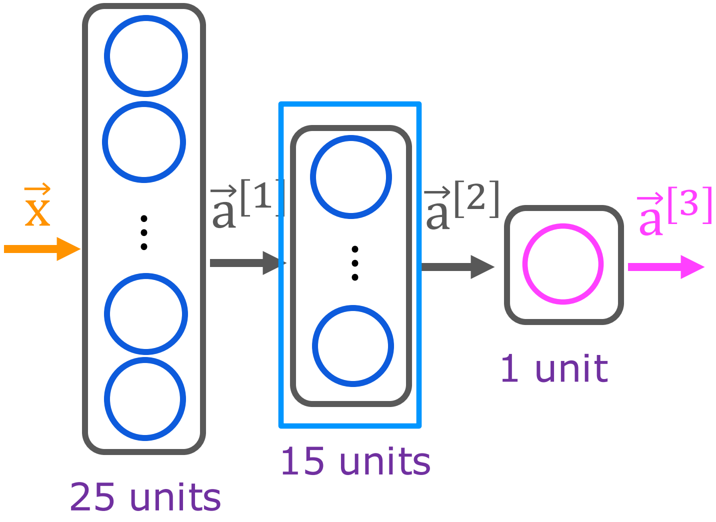
Model for digit classification




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

Andrew Ng

Model for digit classification

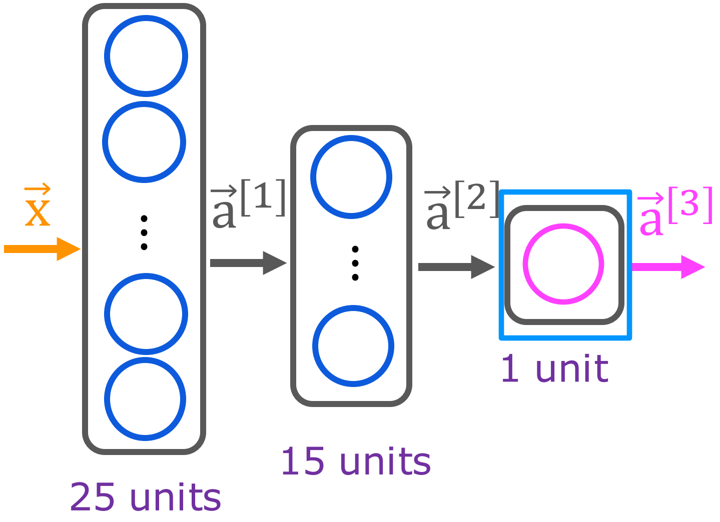



```
x = np.array([[0.0,...245,...240...0]])
layer_1 = Dense(units=25, activation='sigmoid')
a1 = layer_1(x)
```

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

Andrew Ng

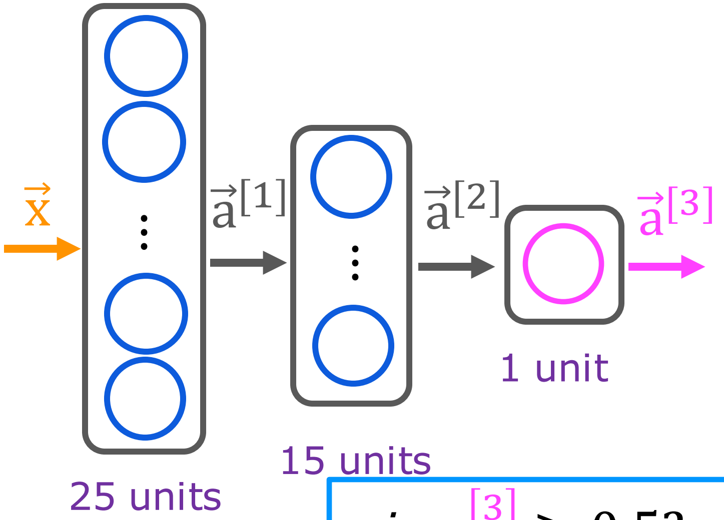
Model for digit classification




```
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)
```

Andrew Ng

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
```

Andrew Ng

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?

Andrew Ng

Note about numpy arrays

$$\text{2 rows}\left\{\overbrace{\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}}^{\text{3 columns}}\right.$$

2 x 3 matrix

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

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

2D array

2 x 3

$$\text{4 rows}\left\{\underbrace{\begin{bmatrix}0.1 & 0.2\\ -3 & -4\\ -.5 & -.6\\ 7 & 8\end{bmatrix}}_{\text{2 columns}}\right.$$

4 x 2 matrix

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

```
[[0.1, 0.2],
 [-3.0, -4.0,],
 [-0.5, -0.6,],
 [7.0, 8.0,]]
```

4 x 2

1 x 2

2 x 1

Andrew Ng

Note about numpy arrays

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

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

1 x 2

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

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

2 x 1

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

1D
"Vector"

Andrew Ng

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]])
```

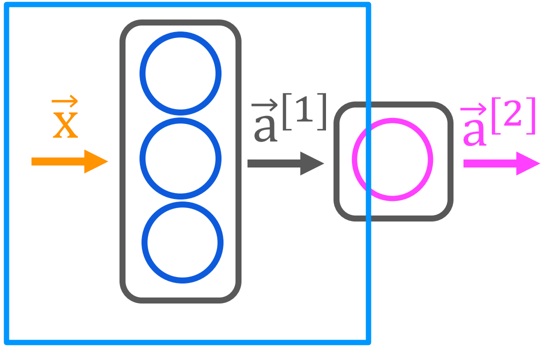
```
[[200.0, 17.0]]
```

1 x 2

$$[200.0 \quad 17.0]$$

Andrew Ng

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

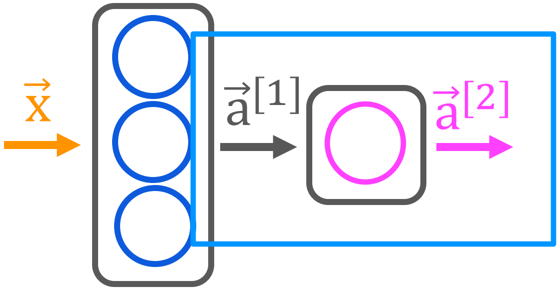
```
tf.Tensor([[0.2 0.7 0.3]], shape=(1, 3), dtype=float32)
```

```
a1.numpy()
```

```
array([[1.4661001, 1.125196 , 3.2159438]], dtype=float32)
```

Andrew Ng

Activation vector




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

[[0.8]]

1 x 1

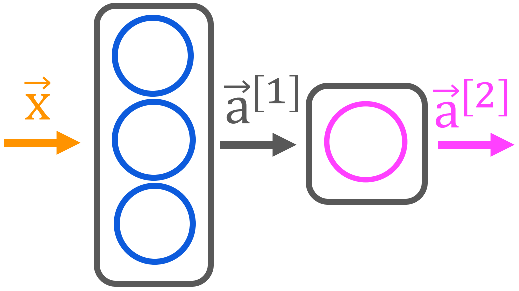
```
tf.Tensor([[0.8]], shape=(1, 1), dtype=float32)
```

```
a2.numpy()
```

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

Andrew Ng

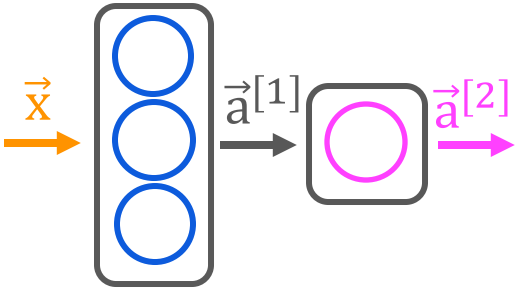
What you saw earlier




```
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)
```

Andrew Ng

Building a neural network architecture



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

layer 1

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

layer 2

```
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]])
```

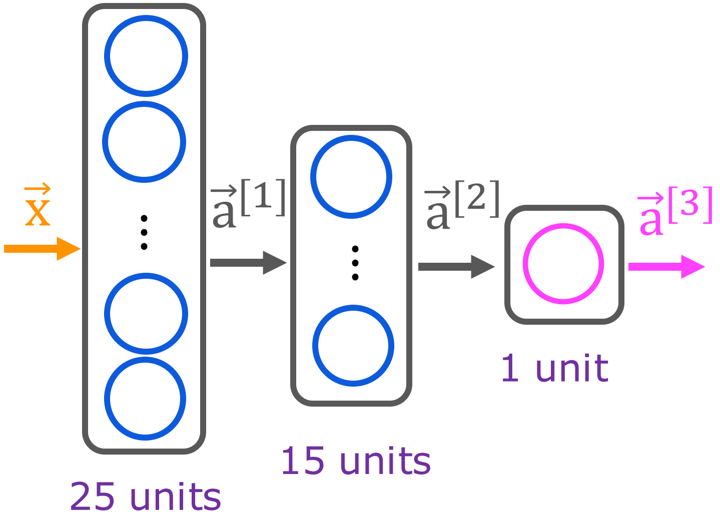
4 x 2

targets

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

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

model.predict(x_new)
```

Andrew Ng

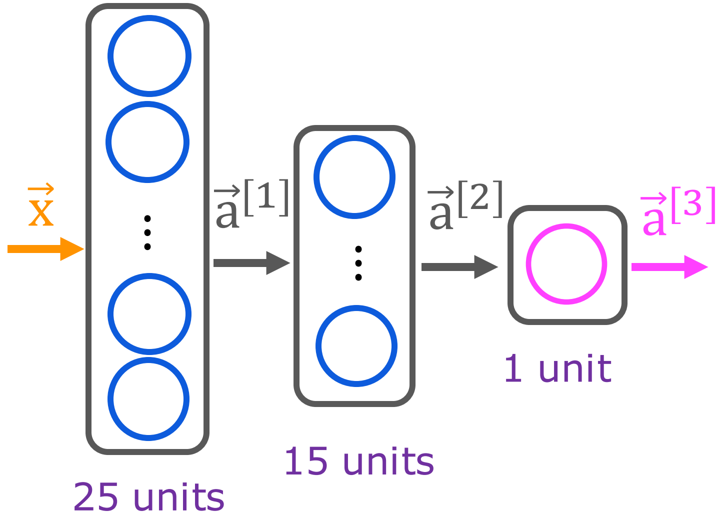
Digit classification model




```
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)
```

Andrew Ng

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)
```

Andrew Ng