MACHINE LEARNING COURSE

PRESENTED BY ABDEL RAHMAN ALSABBAGH

LECTURE #5 – WED – 24.5.2023

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

In the name of Allah, the most gracious, the most merciful, we start :)

Today’s Quote

“Start by doing what’s necessary; then do what’s possible; and suddenly you are doing the impossible”

- Francis of Assisi

Artificial Neural Networks

- ANNs.
- Tensorflow/Keras.
- Forward propagation.
- Activcation functions.

Source: Machine Learning Specialization by Andrew Ng and Stanford Online.

TECH REQUIREMENTS

Artificial Neural Networks

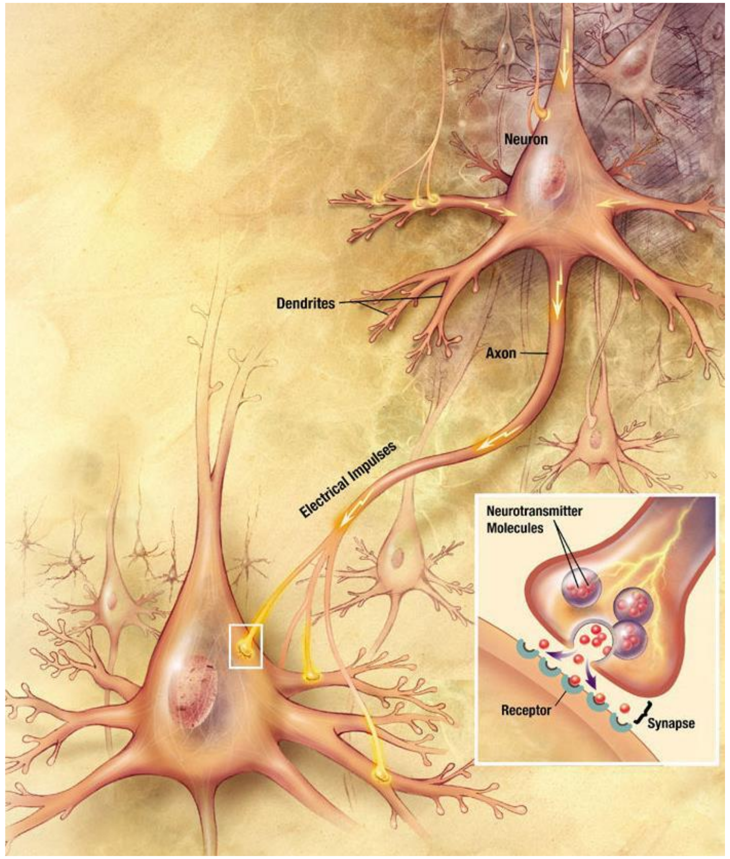
Origins: Algorithms that try to mimic the brain.

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) → ...

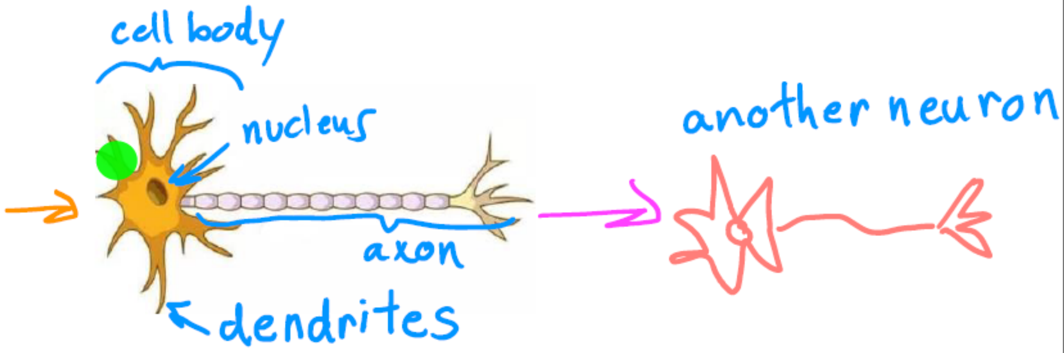
Neural Networks



Neural Networks

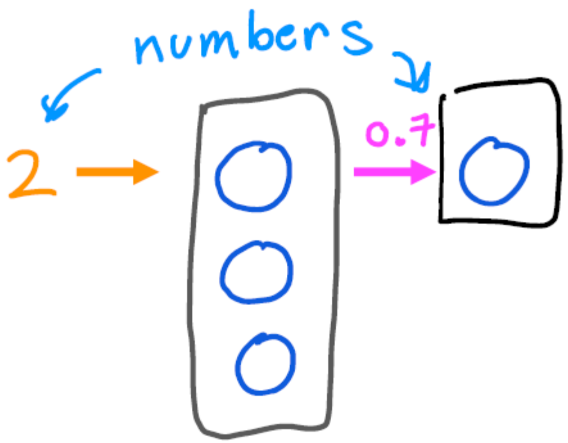
Biological neuron

inputs outputs

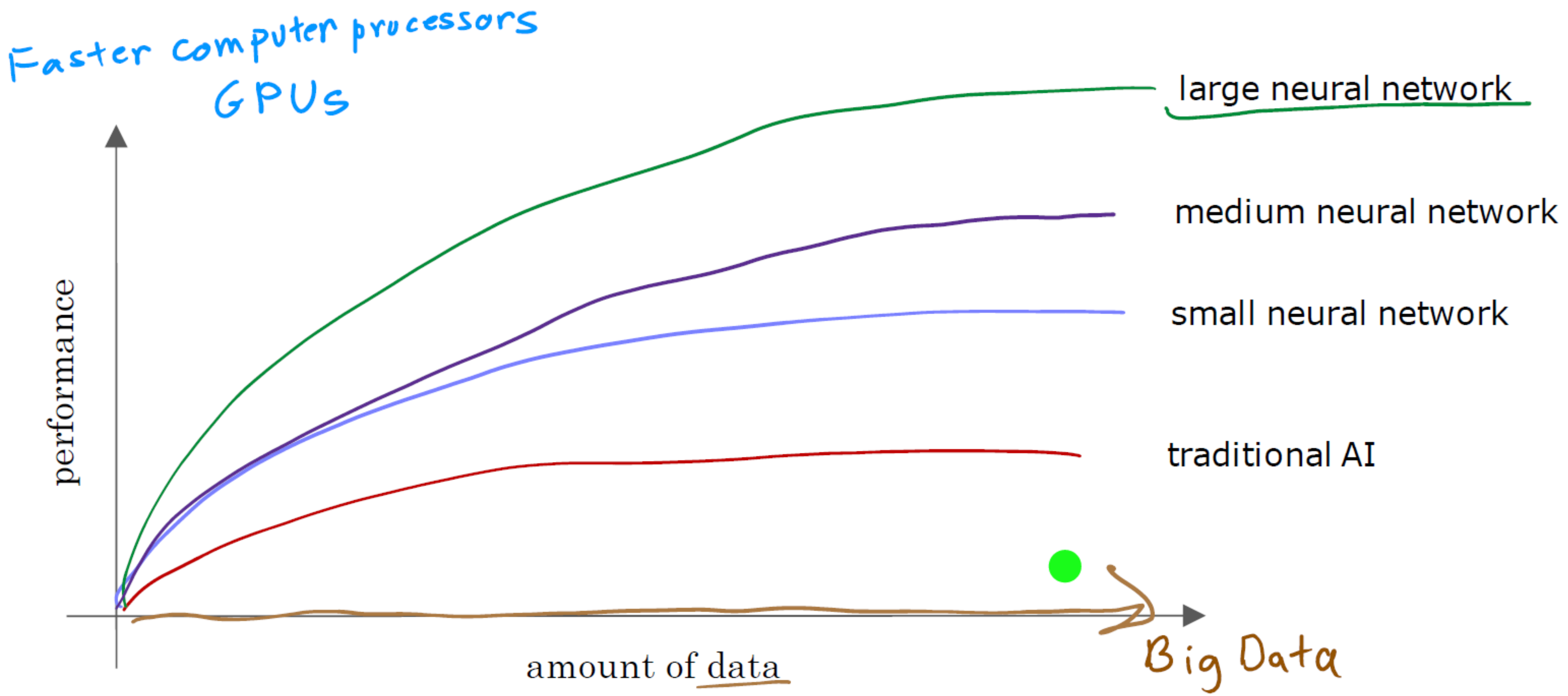


Simplified mathematical model of a neuron

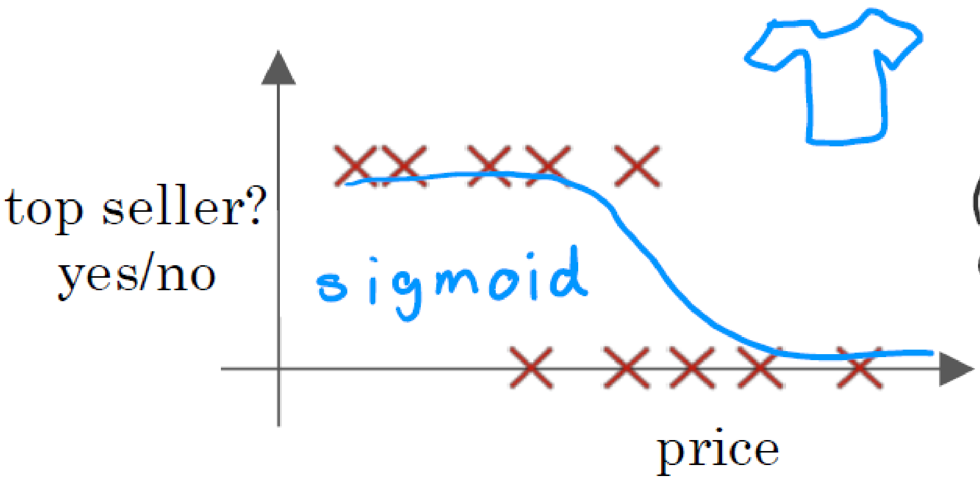
inputs outputs



Why Now?


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

Demand Prediction



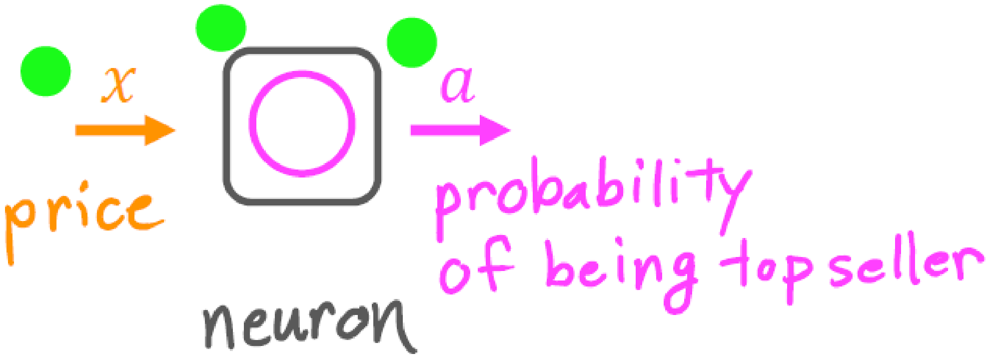
$$x = price$$

input

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

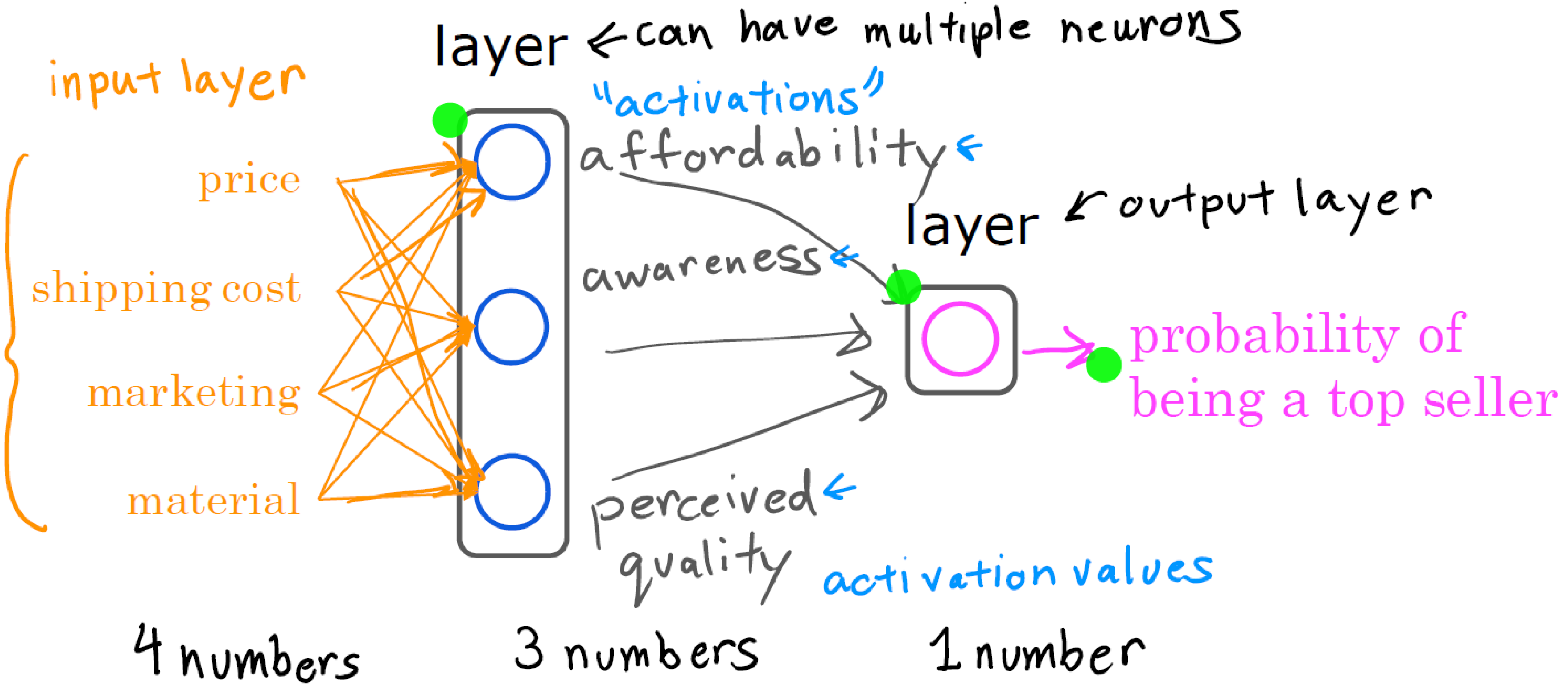
activation

output

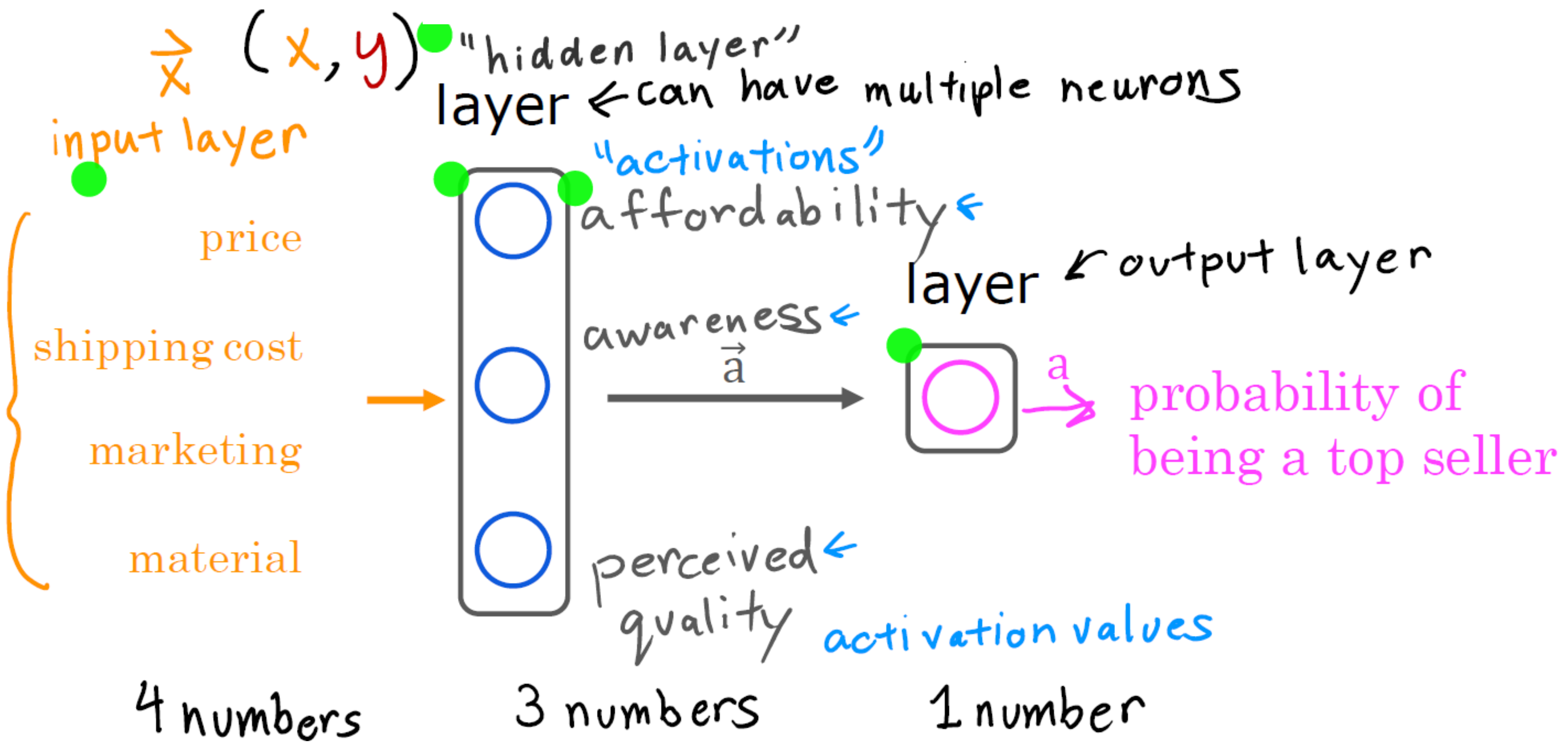


Demand Prediction

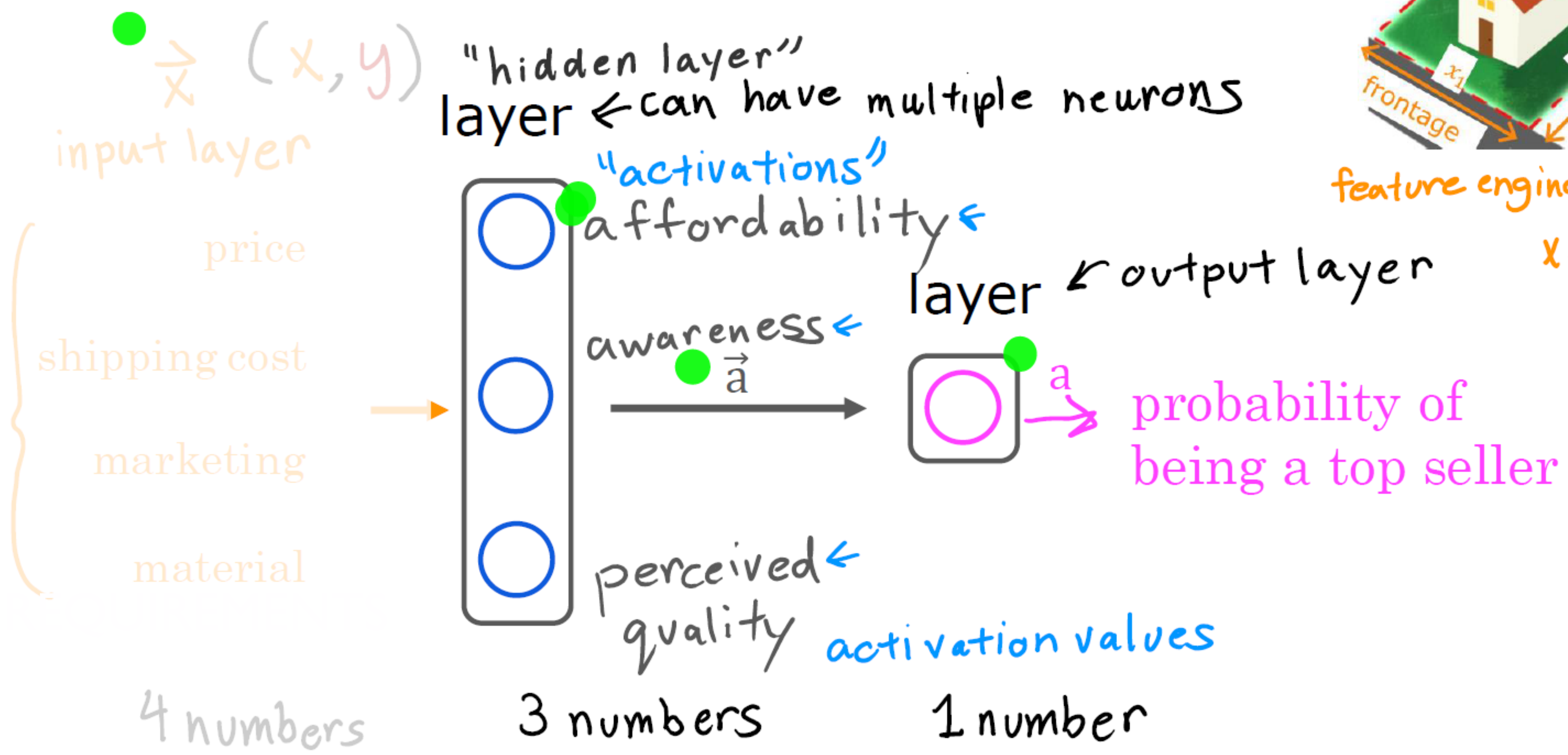
Demand Prediction



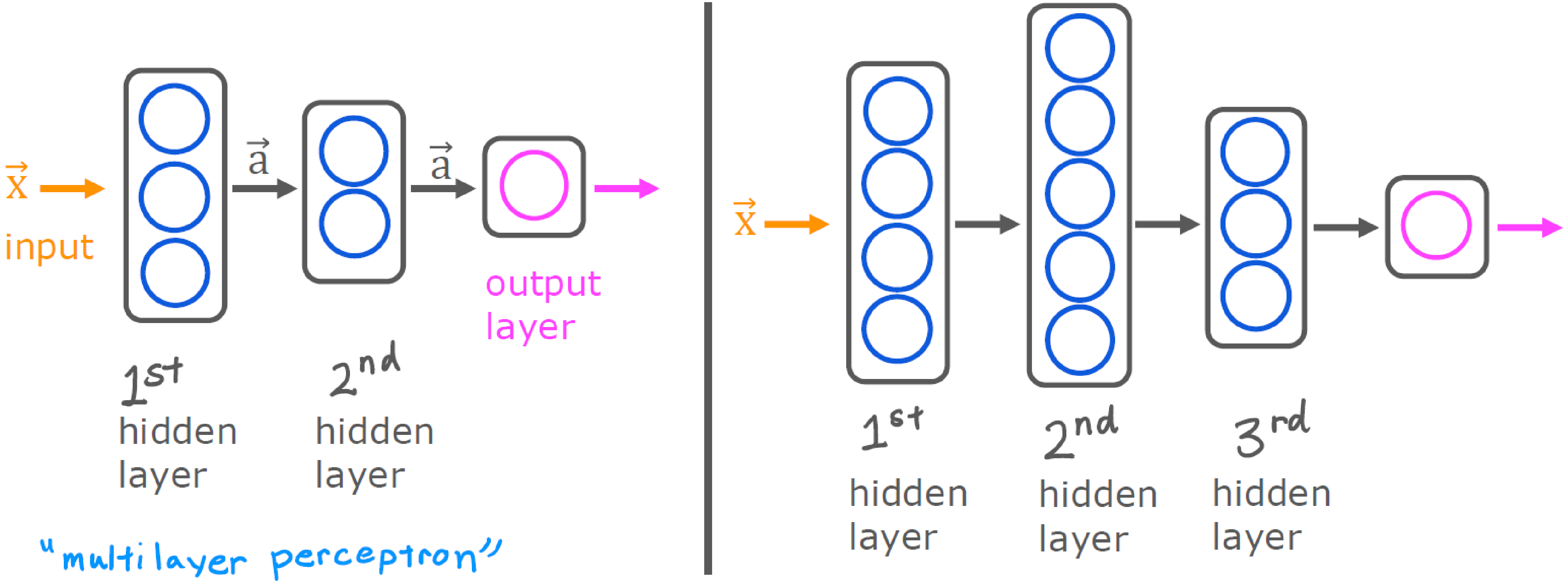
Demand Prediction



Demand Prediction

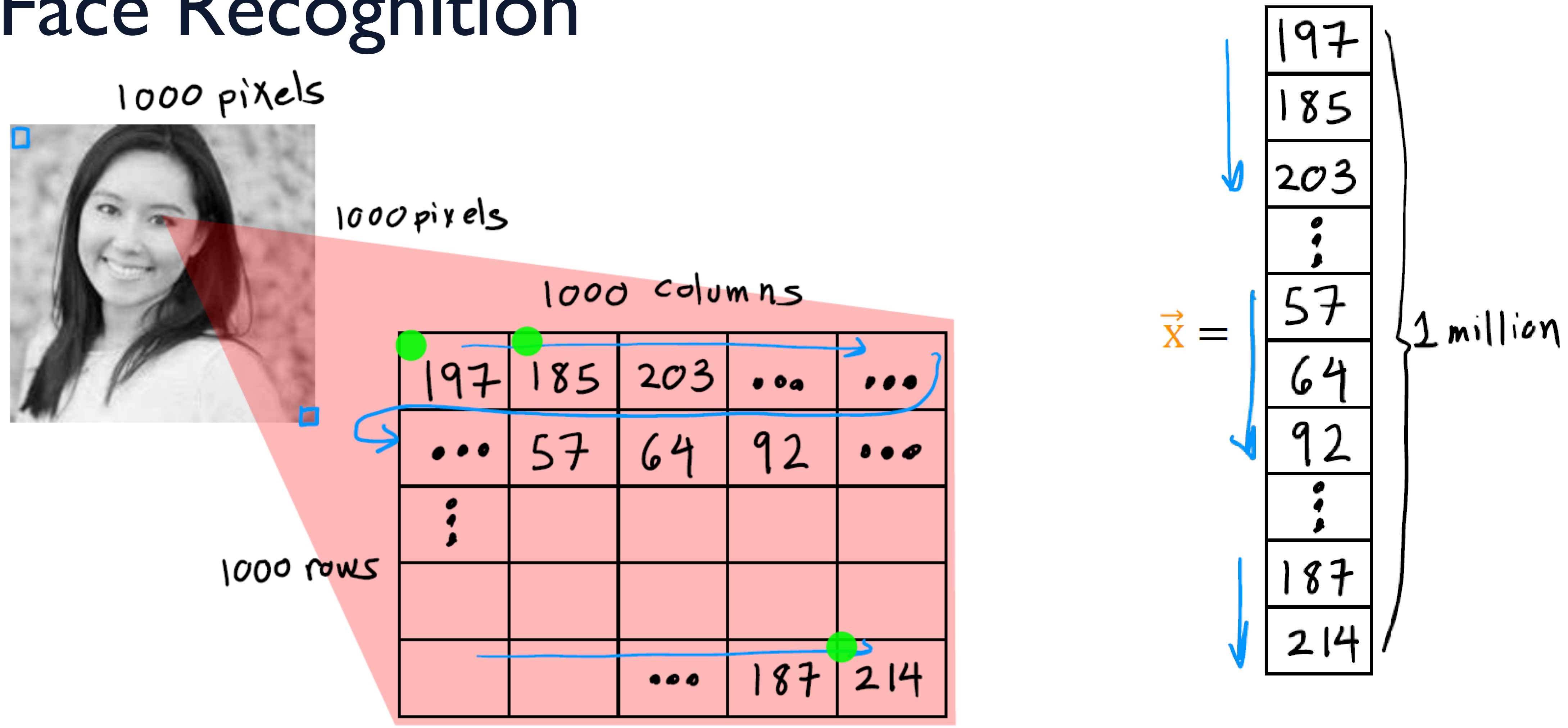


Multiple Hidden Layers

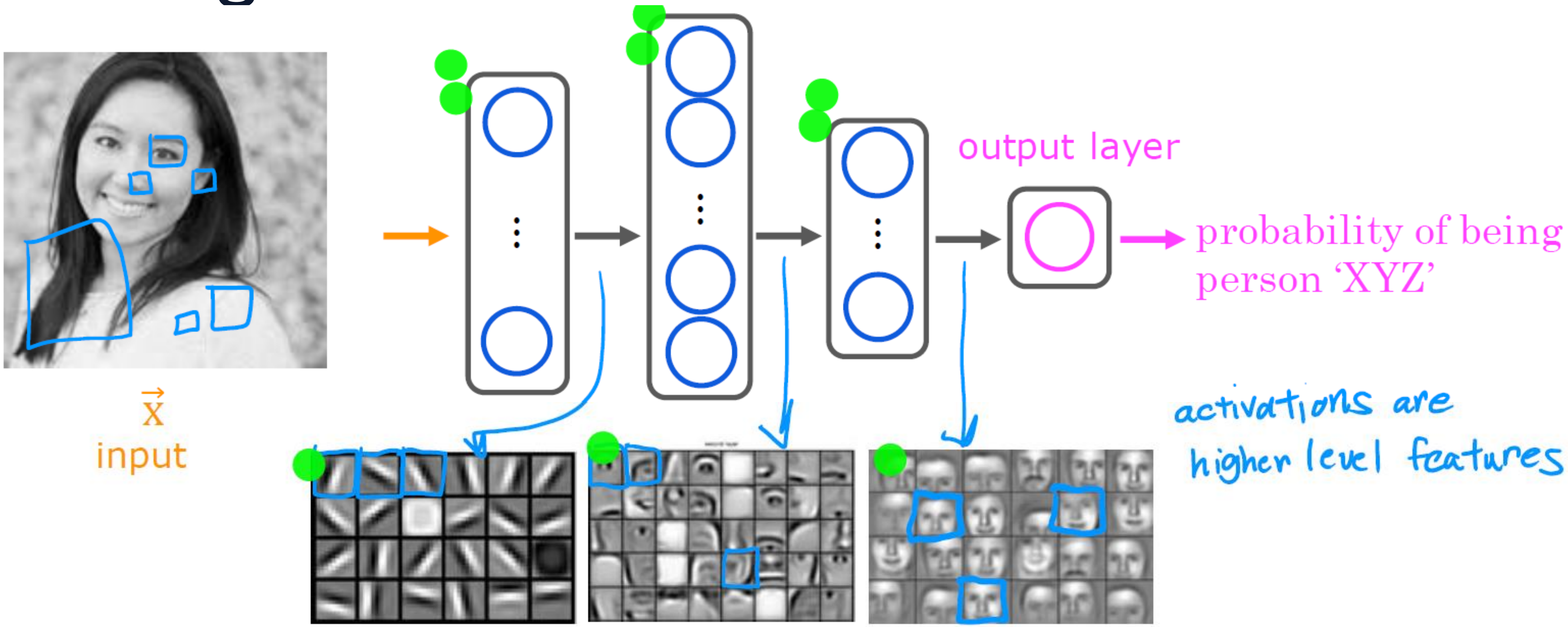


neural network architecture

Face Recognition

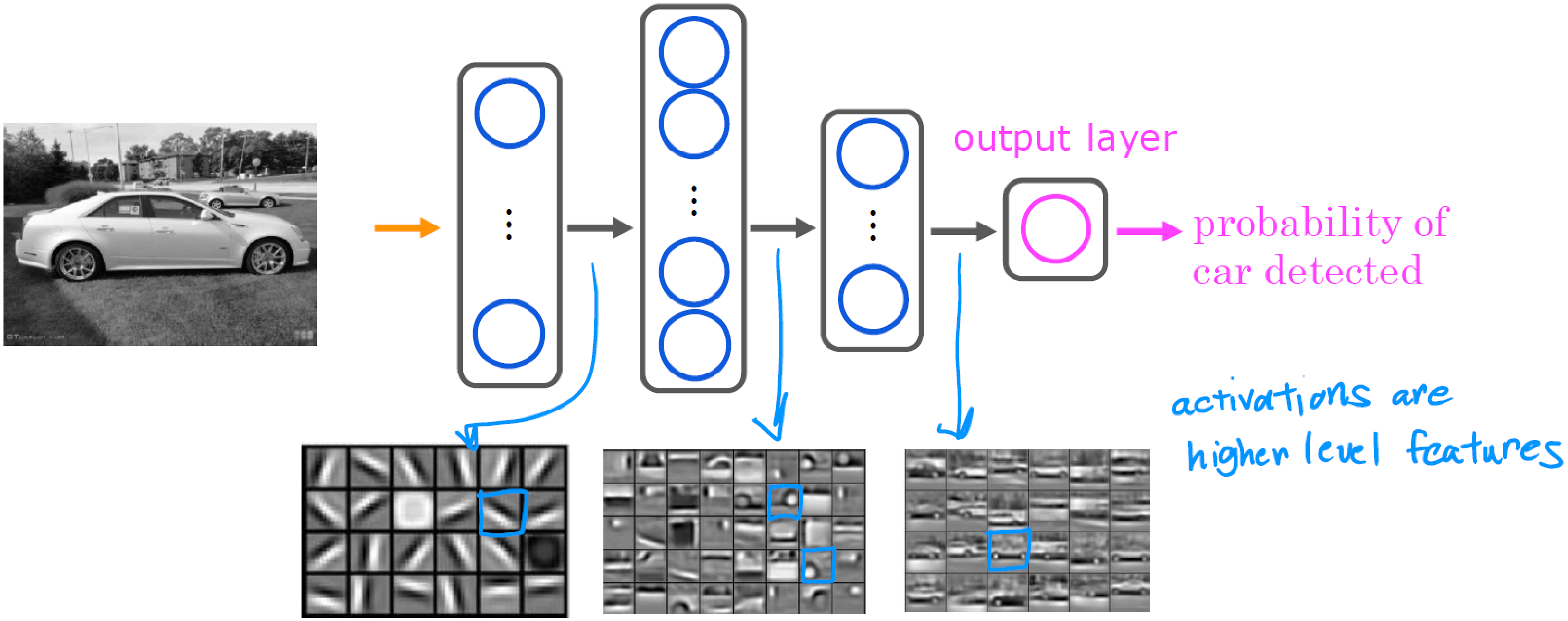


Face Recognition

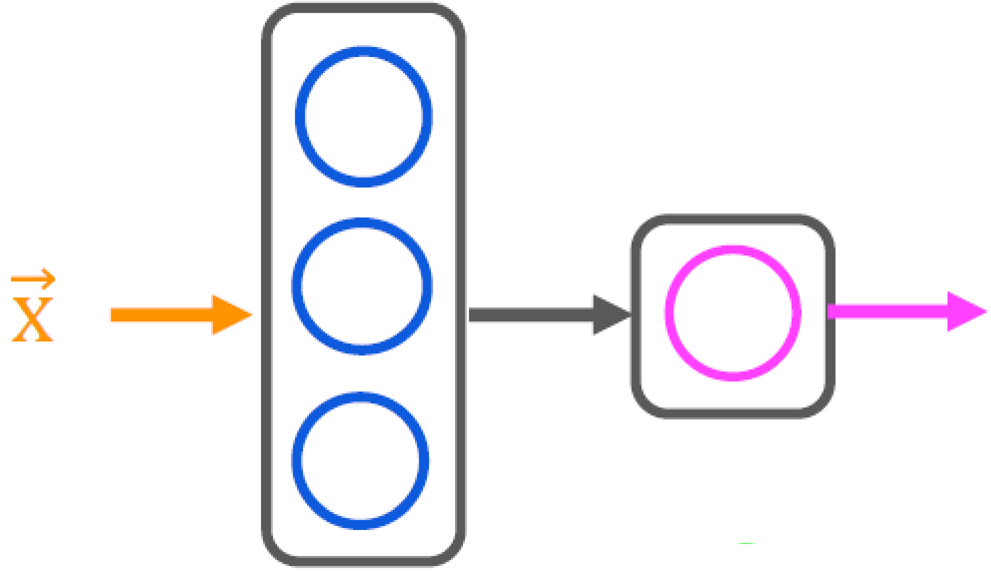


source: Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations by Honglak Lee, Roger Grosse, Ranganath Andrew Y. Ng

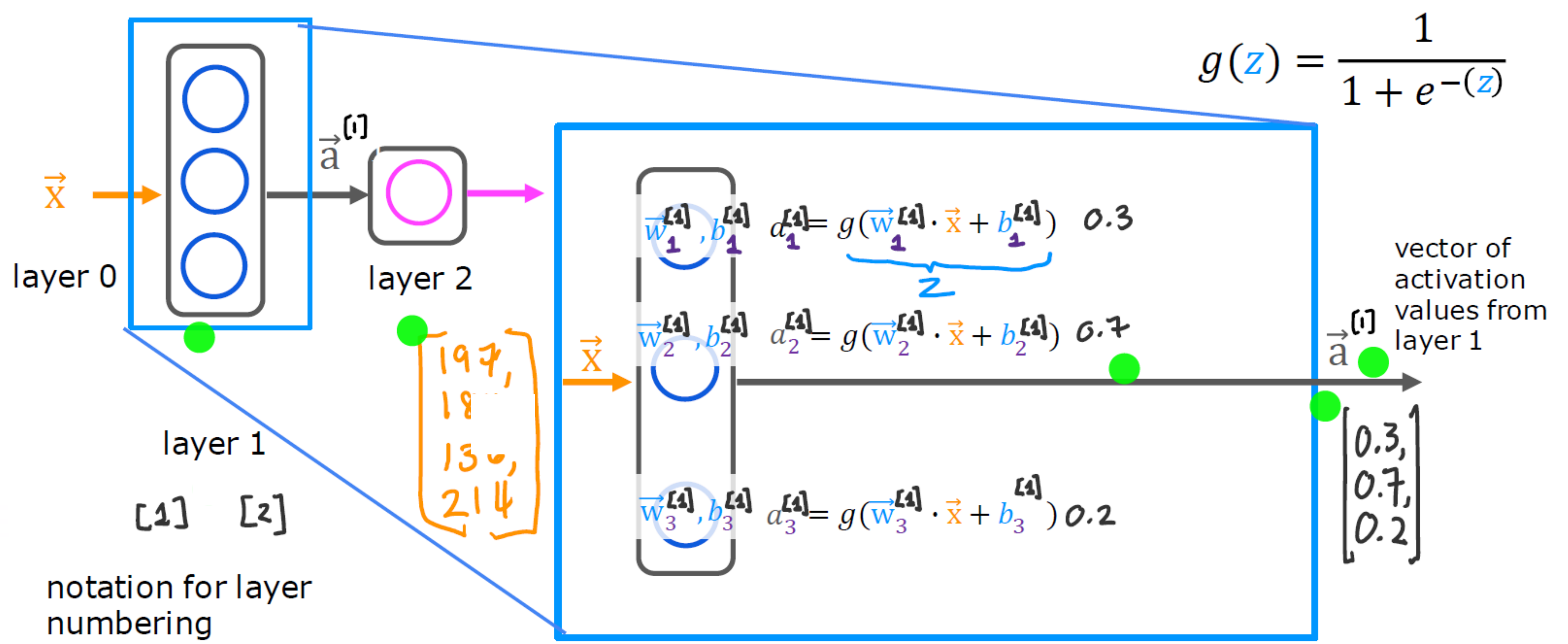
Car Classification



Neural Network Layer

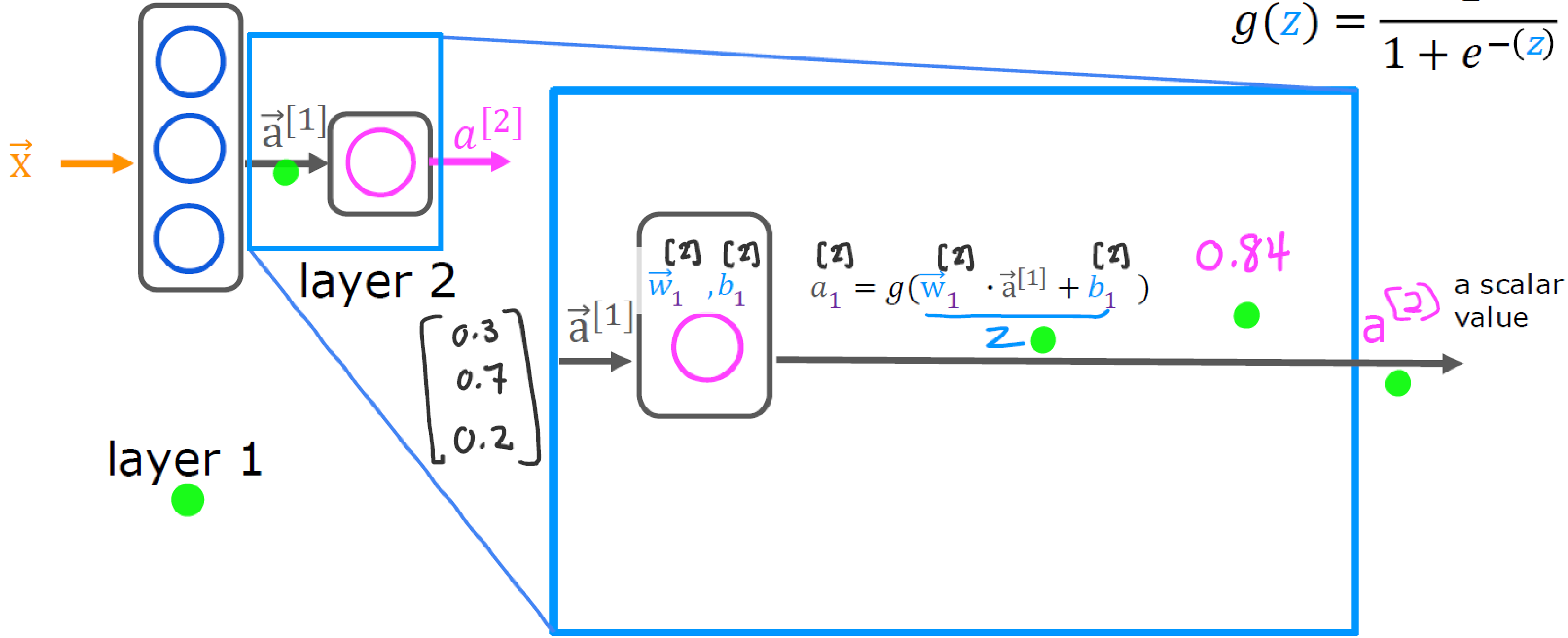


Neural Network Layer

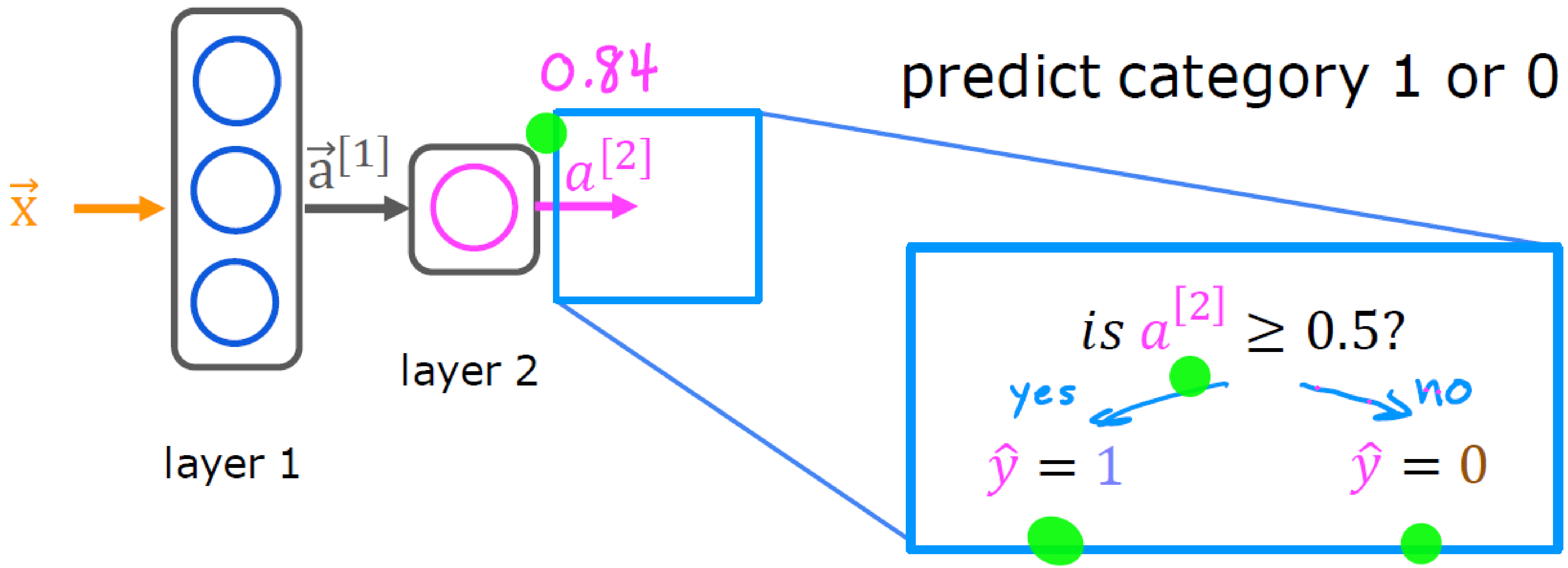


Neural Network Layer

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

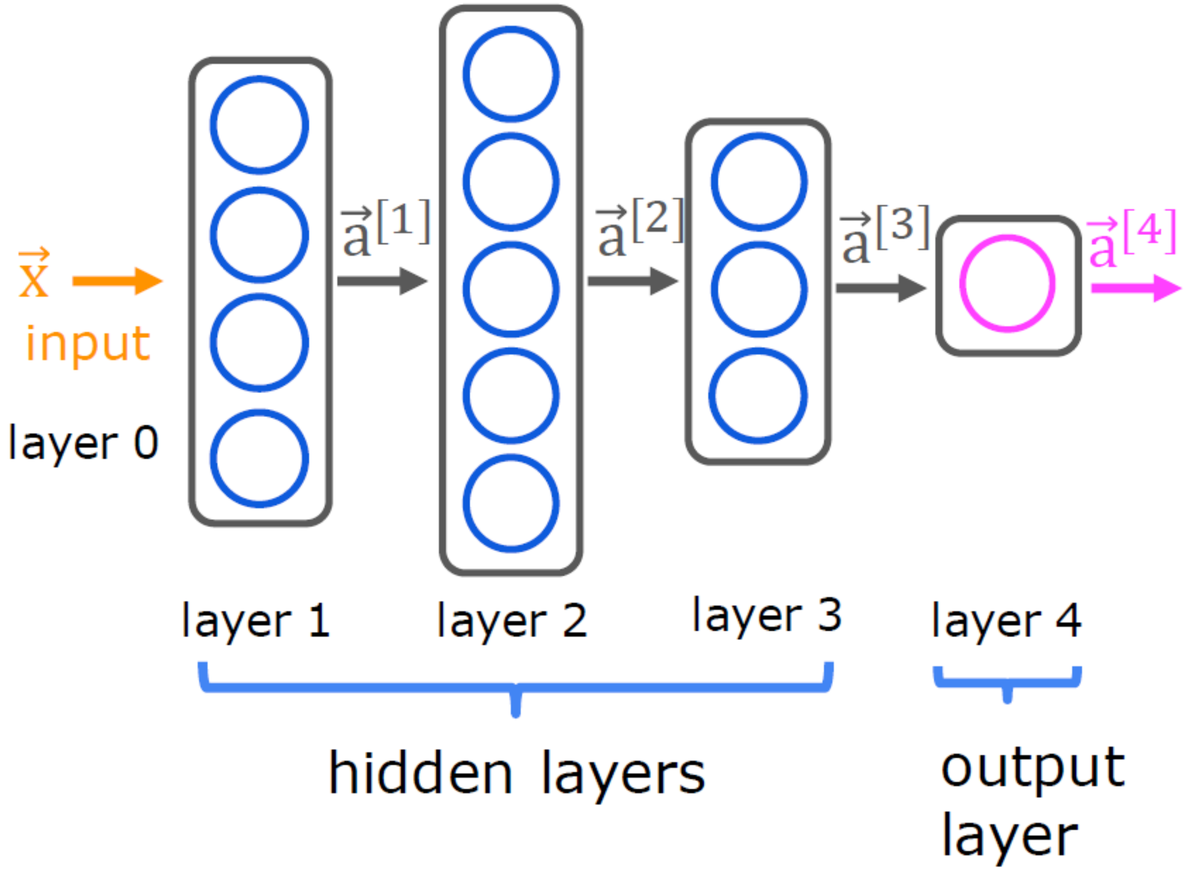


Neural Network Layer

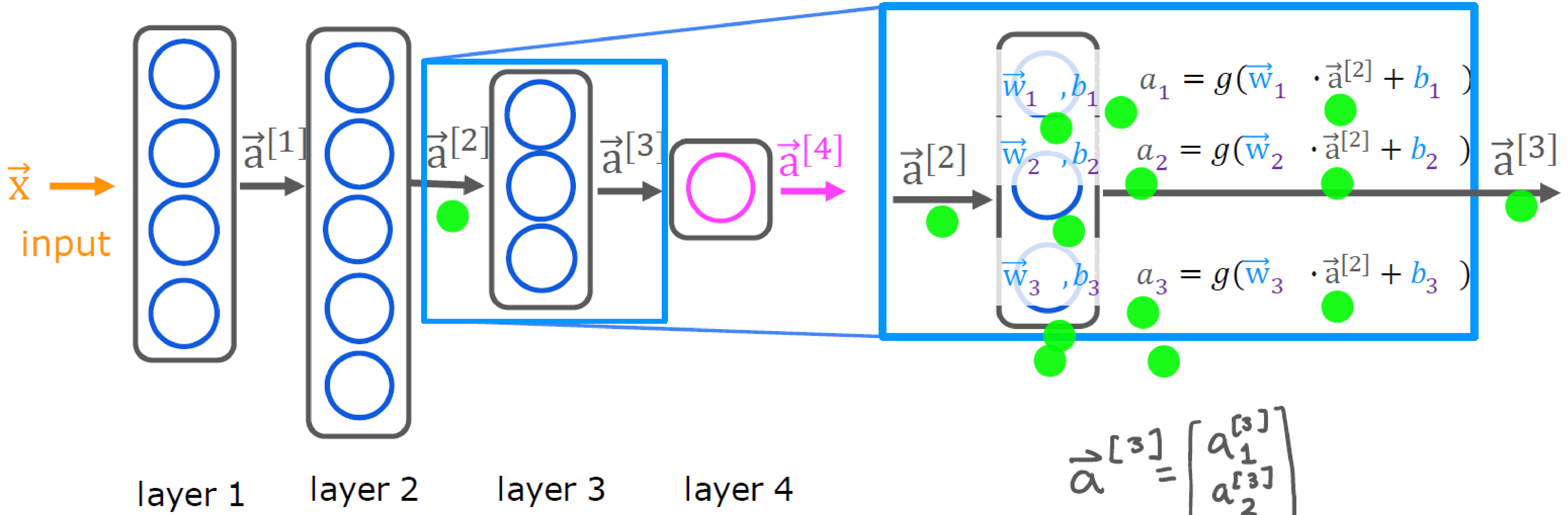


predict category 1 or 0 (yes/no)

More Complex Neural Network

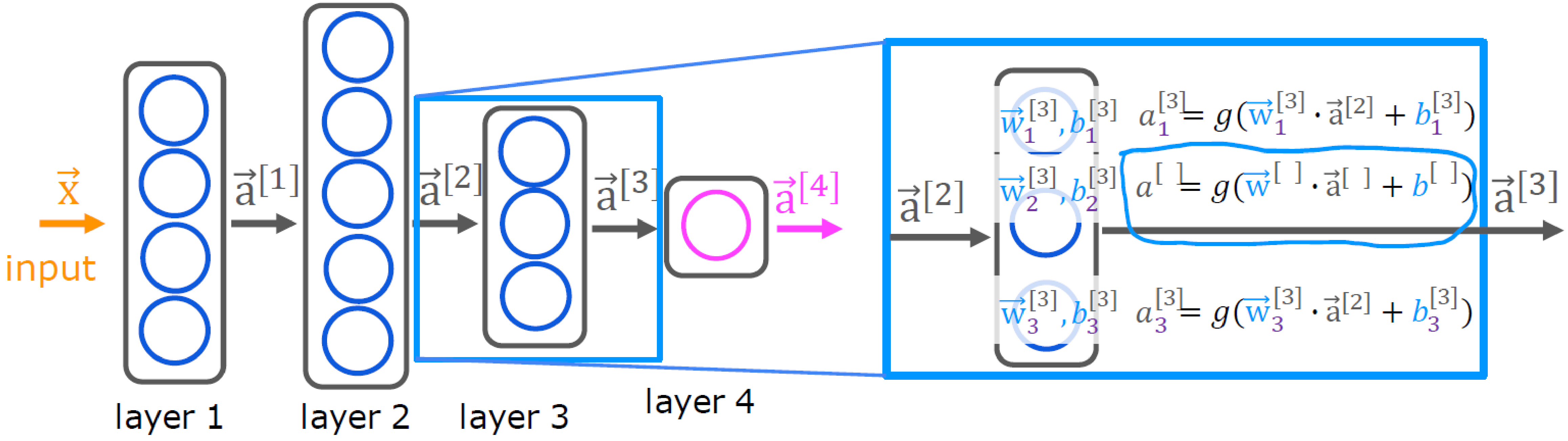


More Complex Neural Network



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

More Complex Neural Network



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

More Complex Neural Network

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

Activation value of layer l, unit(neuron) j

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

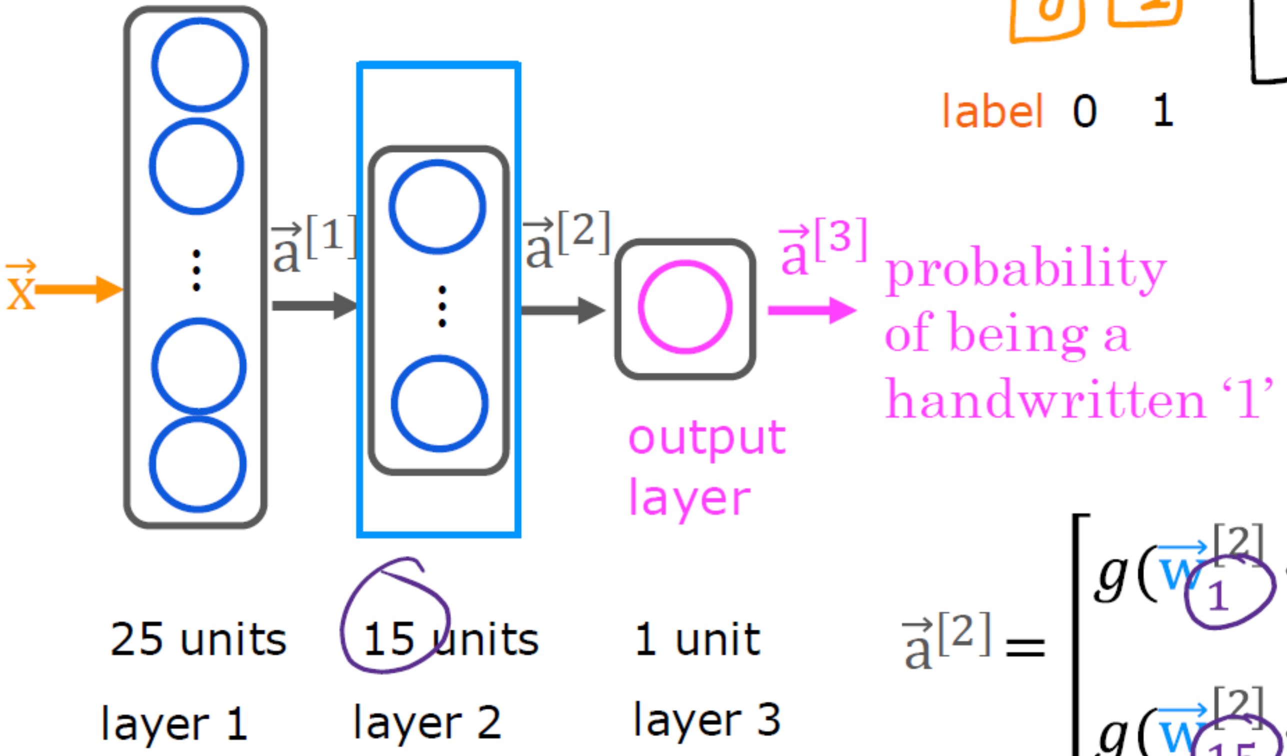
$$\boxed{a_j^{[l]} = g(\vec{w}_j^{[l]} \cdot \vec{a}^{[l-1]} + b_j^{[l]})}$$

sigmoid

"activation function"

Parameters w & b of layer l, unit j

Handwritten Digit Recognition

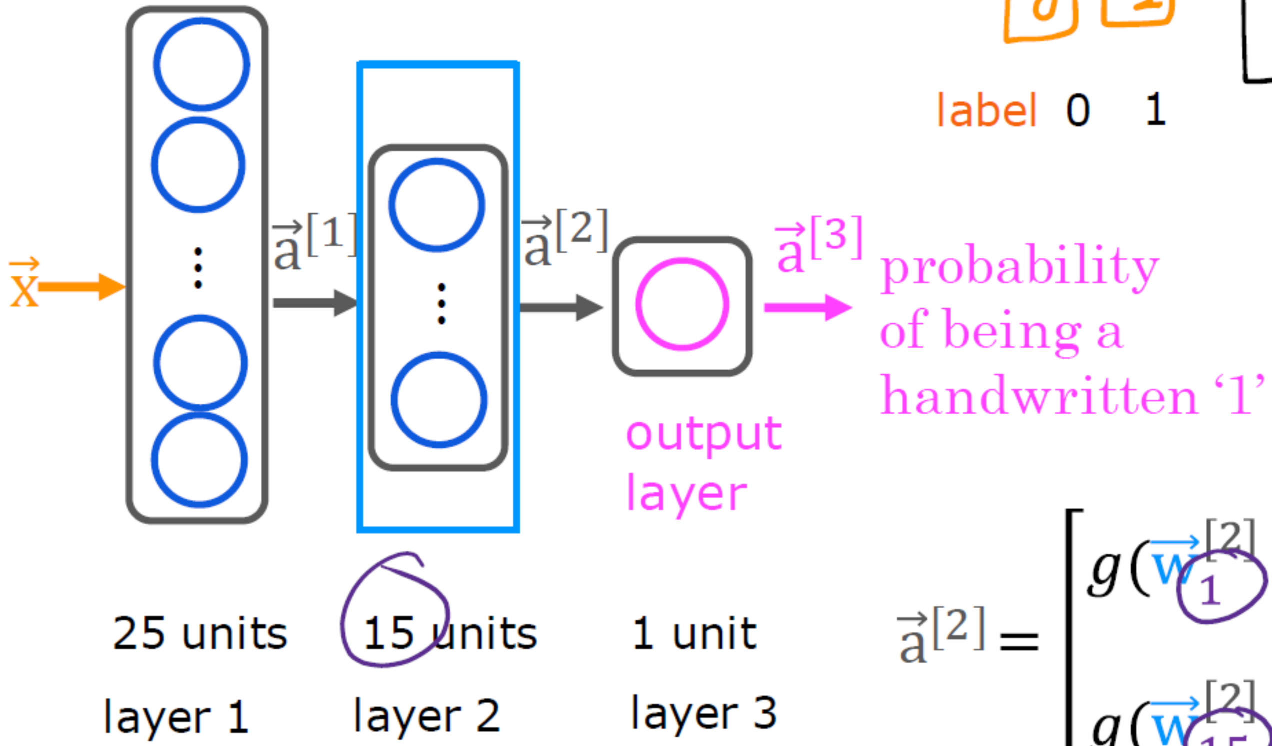


8 × 8

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

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

Handwritten Digit Recognition

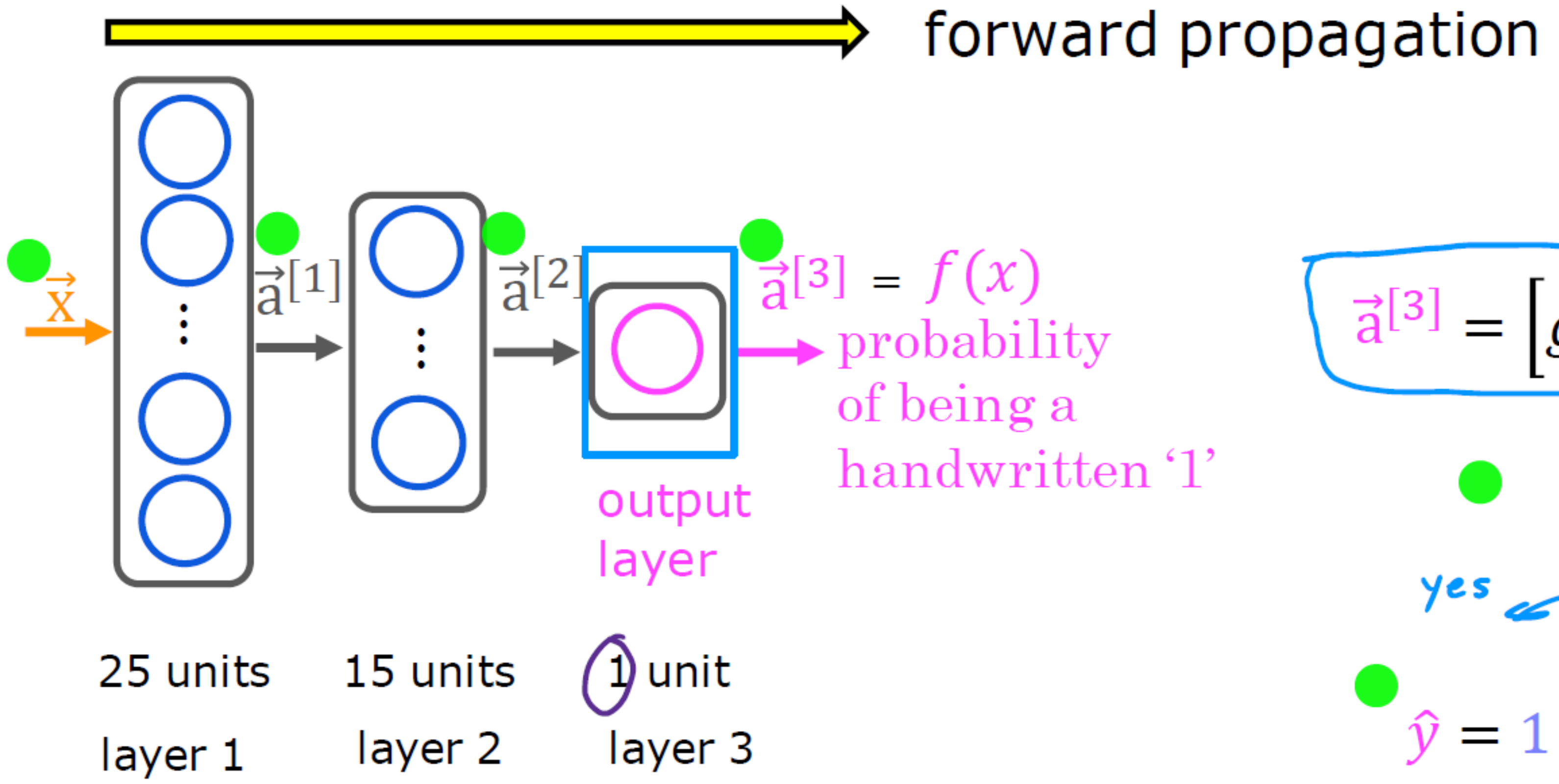


8 × 8

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

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

Forward Propagation



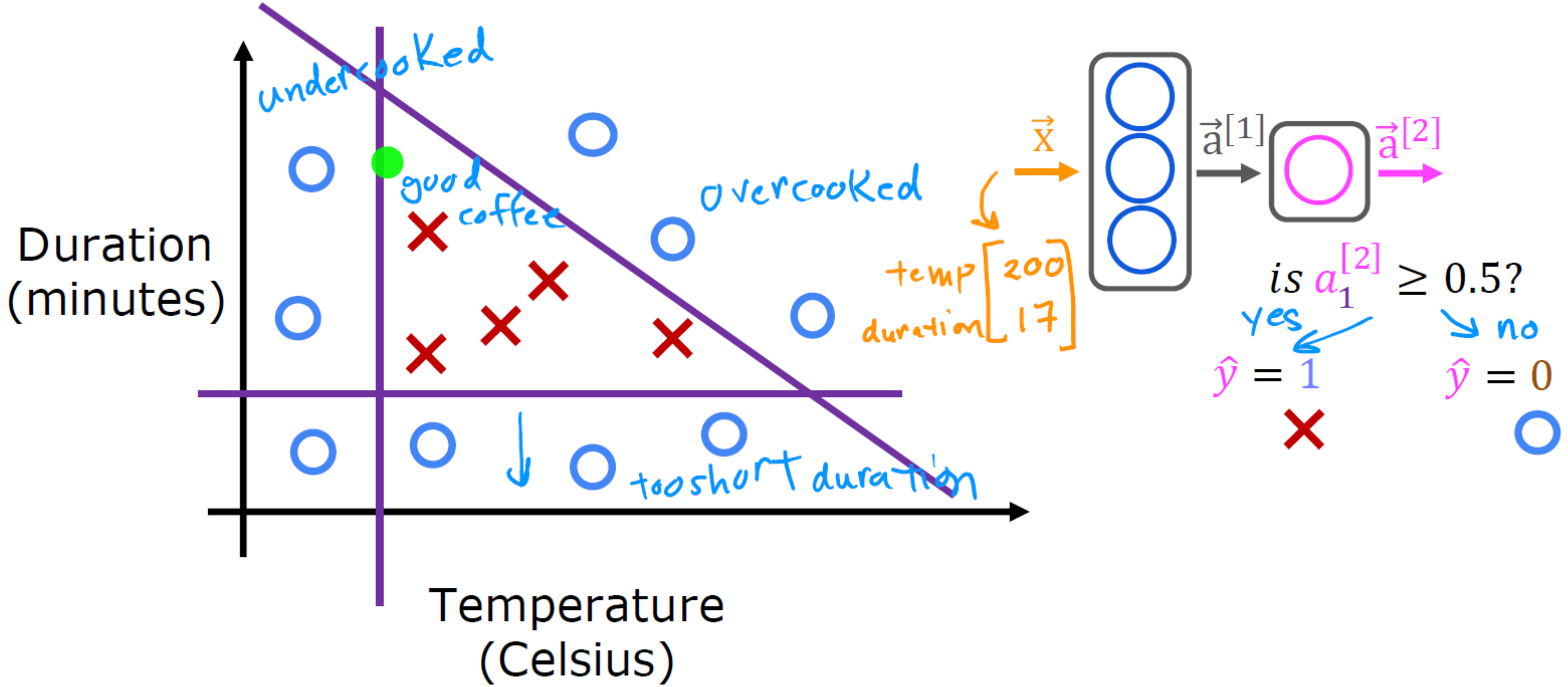
$$\boxed{\vec{a}^{[3]} = \left[g\left(\vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]} \right) \right]}$$

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

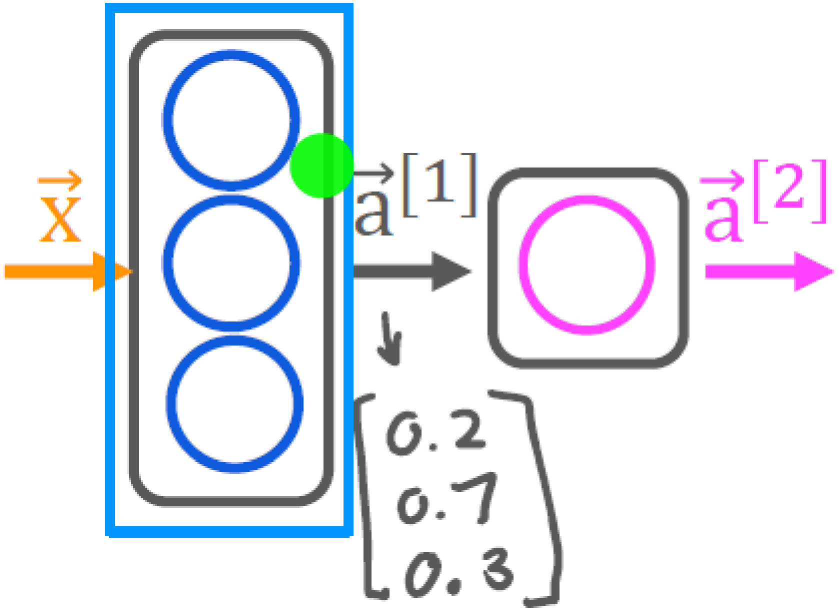
yes — $\hat{y} = 1$ — image is digit 1

no — $\hat{y} = 0$ — image isn't digit 1

Code Example



Build The Model Using TensorFlow/Keras

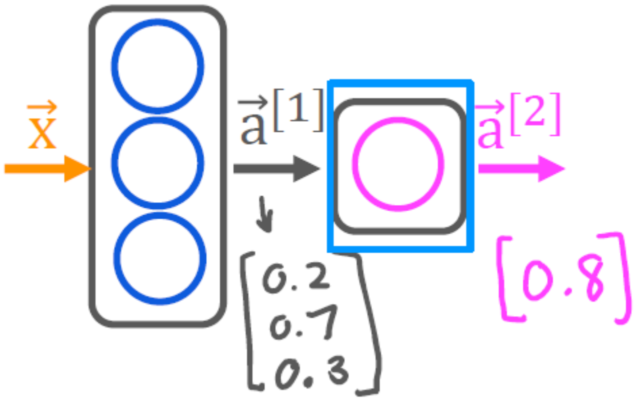



```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation=‘sigmoid’)
a1 = layer_1(x)
```

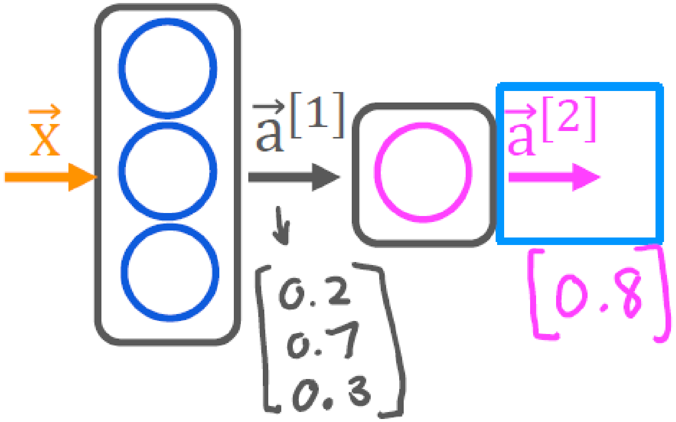
Build The Model Using TensorFlow/Keras

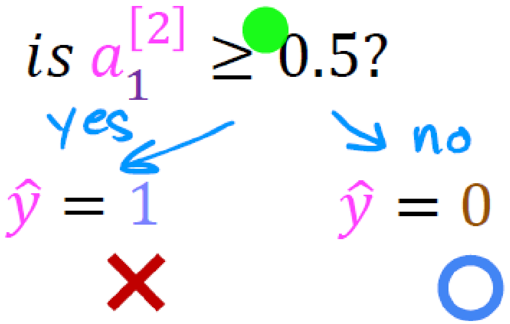
```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation='sigmoid')
a1 = layer_1(x)
```

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



Build The Model Using TensorFlow/Keras



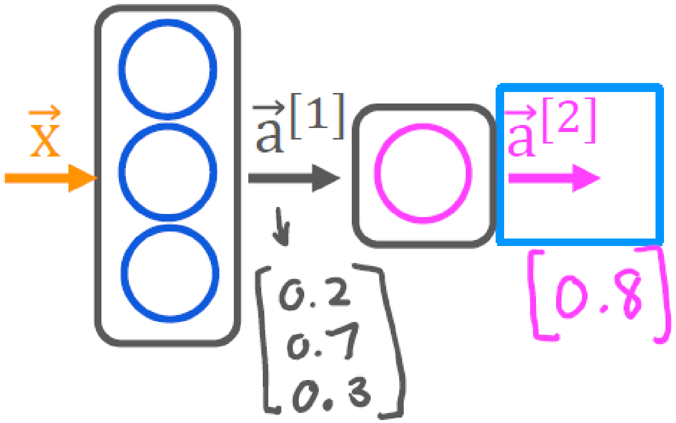


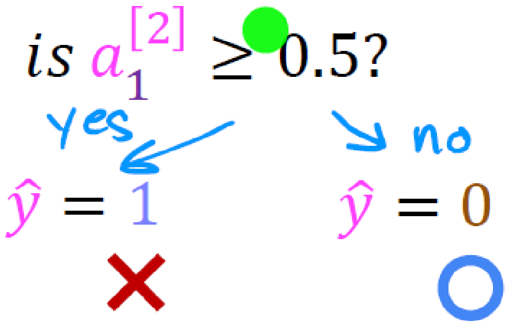

```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation=‘sigmoid’)
a1 = layer_1(x)
```

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

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

Build The Model Using TensorFlow/Keras

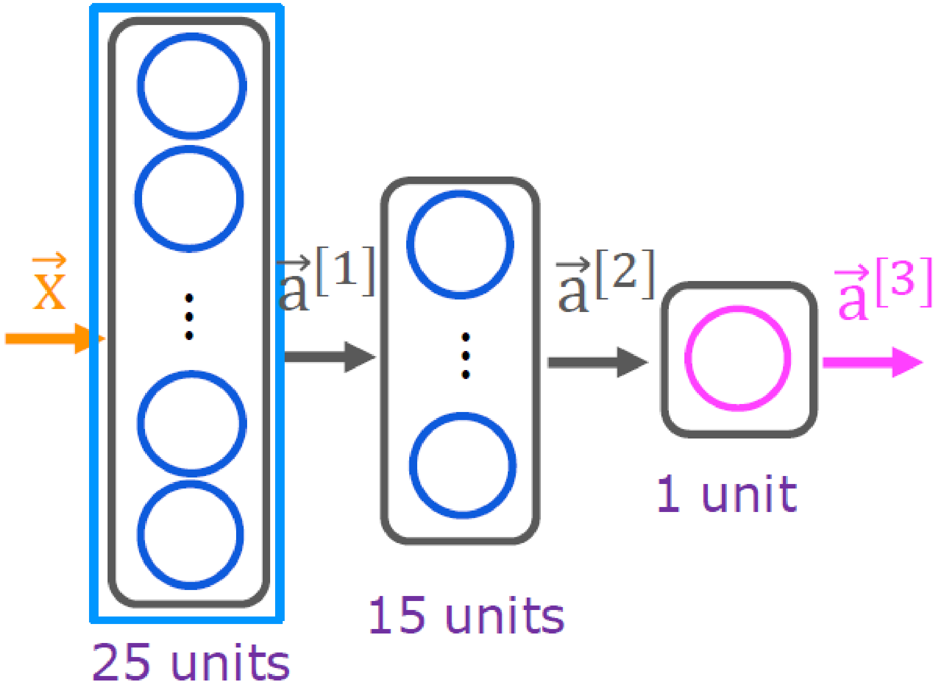





```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation='sigmoid')
a1 = layer_1(x)


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

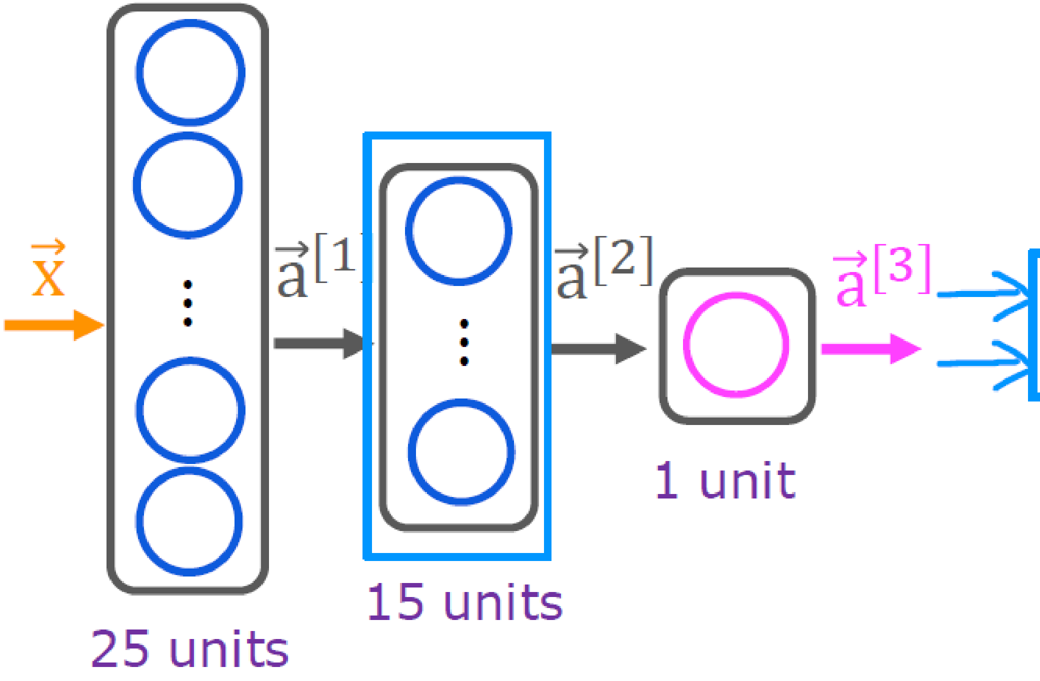
```
if a2 >= 0.5:
  yhat = 1
else:
  yhat = 0
```

Build The Model Using TensorFlow/Keras




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

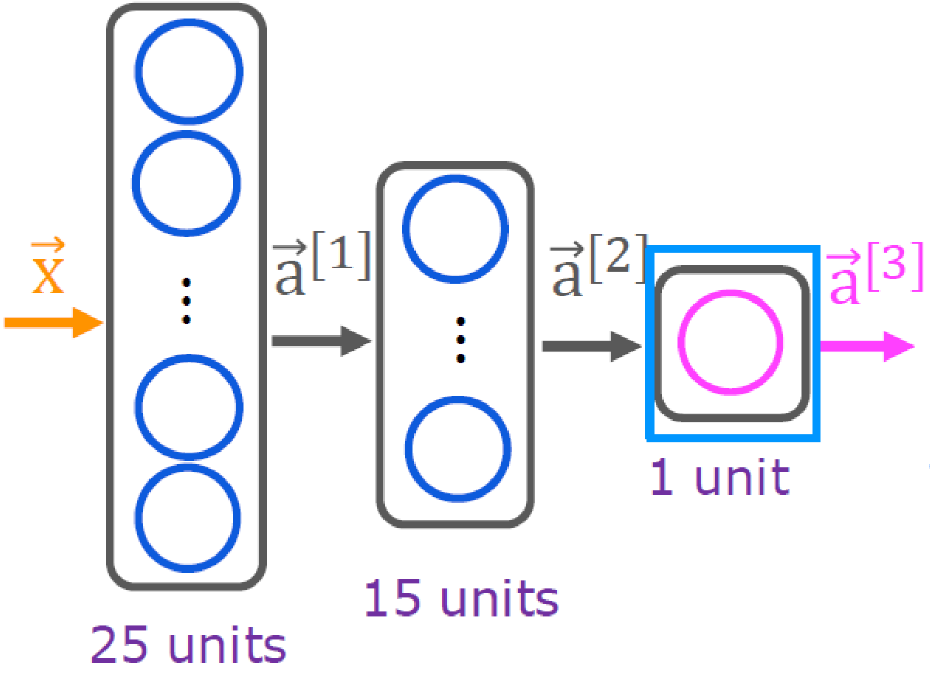
Build The Model Using TensorFlow/Keras




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

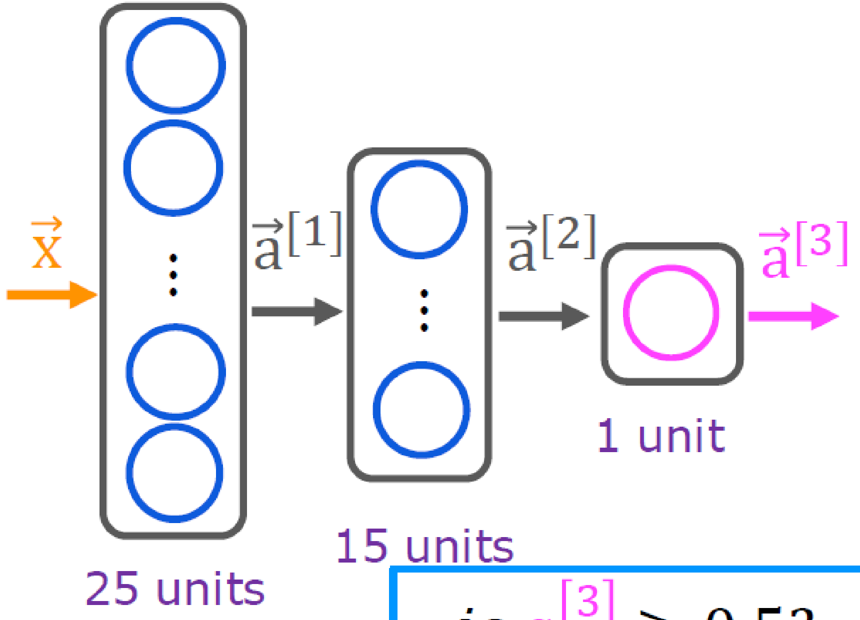
```
layer_2 = Dense(units=15, activation='sigmoid')
a2 = layer_2(a1)
```

Build The Model Using TensorFlow/Keras




```
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)
```

Build The Model Using TensorFlow/Keras




```
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/Keras

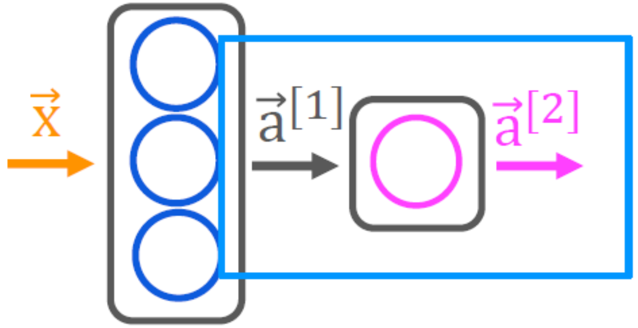
temperature (Celsius)	duration (minutes)	Good coffee? (1/0)
200.0	17.0	1
425.0	18.5	0
...	...	...

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

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

why?

Data in TensorFlow/Keras




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

[[0.8]]

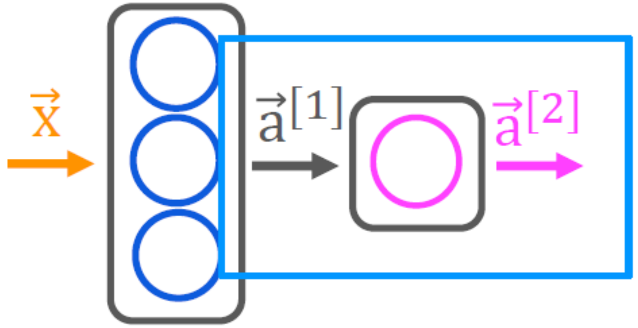
1 x 1

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

```
a2.numpy()
```

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

Activation Vectors




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

[[0.8]]

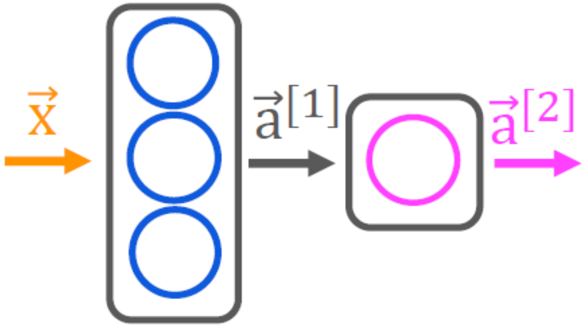
1 x 1

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

```
a2.numpy()
```

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

Building a Neural Network



layer1

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

layer2

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

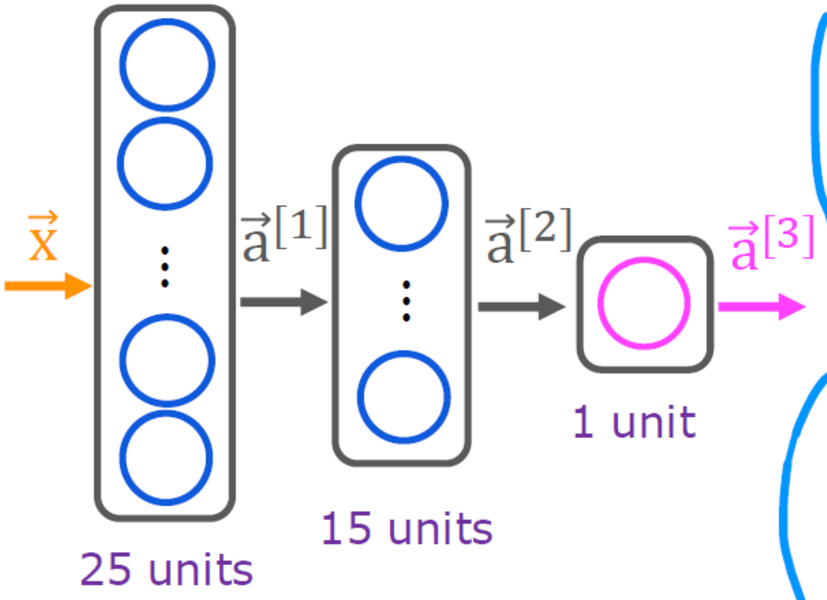
```
x = np.array([[200.0, 17.0],
              [120.0, 5.0],
              [425.0, 20.0],
              [212.0, 18.0]])
```

4 x 2

targets

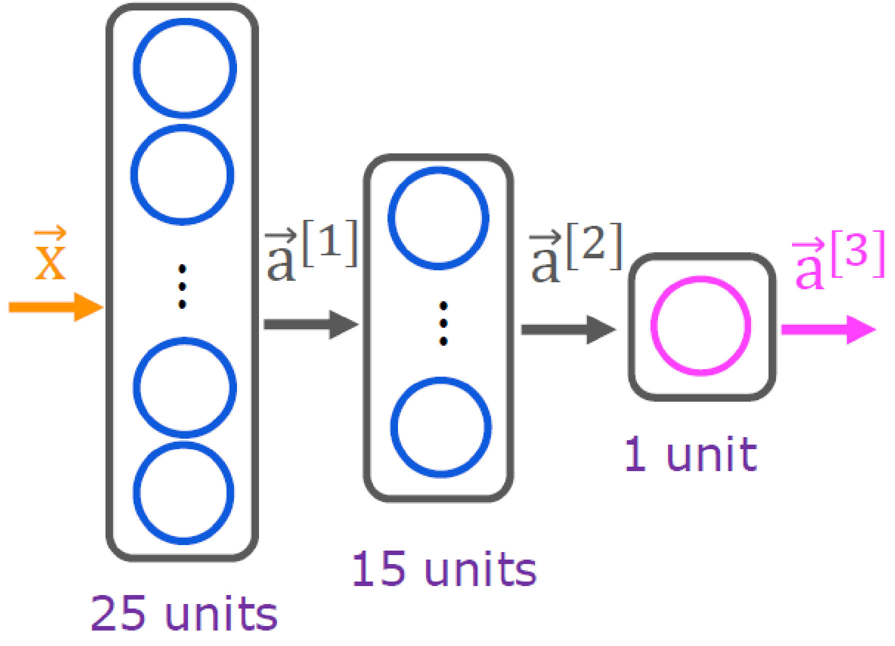
```
y = np.array([1,0,0,1])
model.compile(...)
model.fit(x,y)

model.predict(x_new)
```

Building a Neural Network



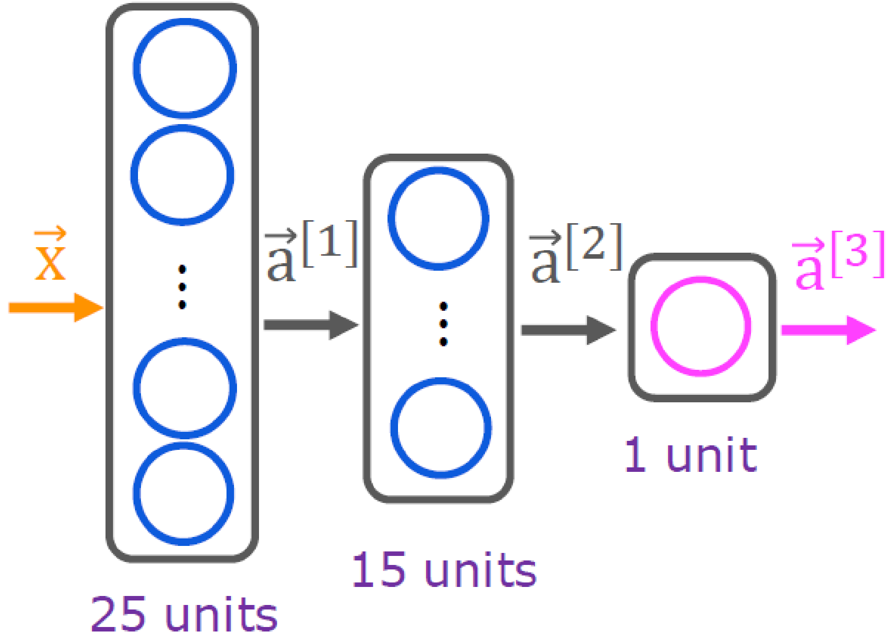

```
layer_1 = Dense(units=25, activation="sigmoid")
layer_2 = Dense(units=15, activation="sigmoid")
layer_3 = Dense(units=1, activation="sigmoid")
model = Sequential([layer_1, layer_2, layer_3]
model.compile(...)
x = np.array([[0..., 245, ..., 17],
              [0..., 200, ..., 184])
y = np.array([1,0])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

Building a Neural Network



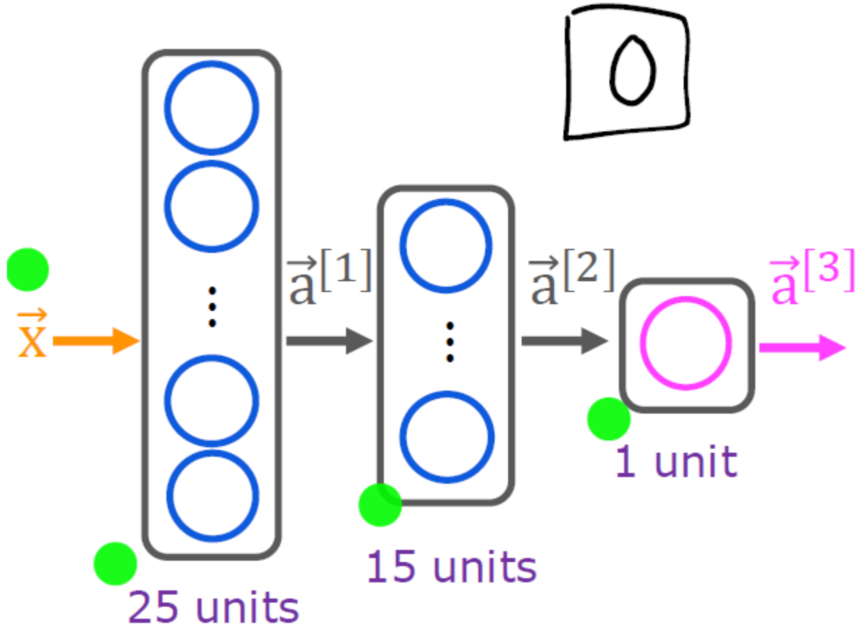

```
model = Sequential([
    Dense(units=25, activation="sigmoid"),
    Dense(units=15, activation="sigmoid"),
    Dense(units=1, activation="sigmoid")])

model.compile(...)

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

model.fit(x,y)

model.predict(x_new)
```

Building a Neural Network




```
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)
```

Building a Neural Network – Details



Given set of (x,y) examples

How to build and train this in code?

```
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

Building a Neural Network – Details

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

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

Keras

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

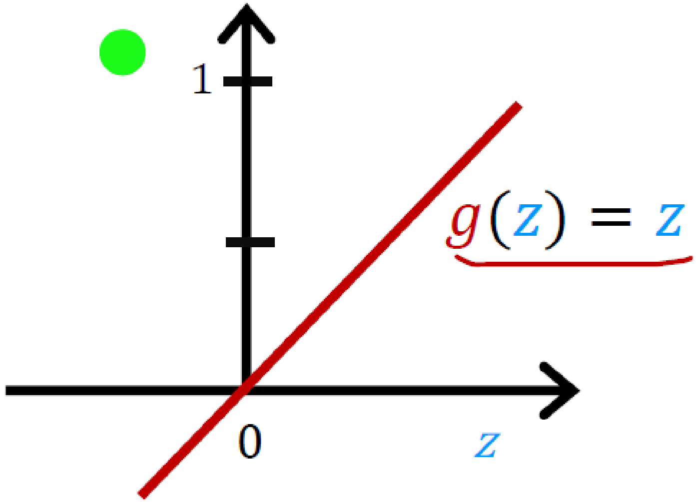
```
from tensorflow.keras.losses import
  MeanSquaredError
```

Activation Functions

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

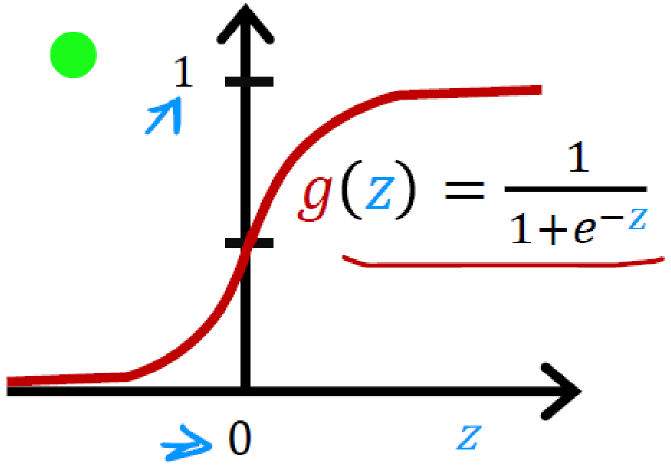
"No activation function"

Linear activation function



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

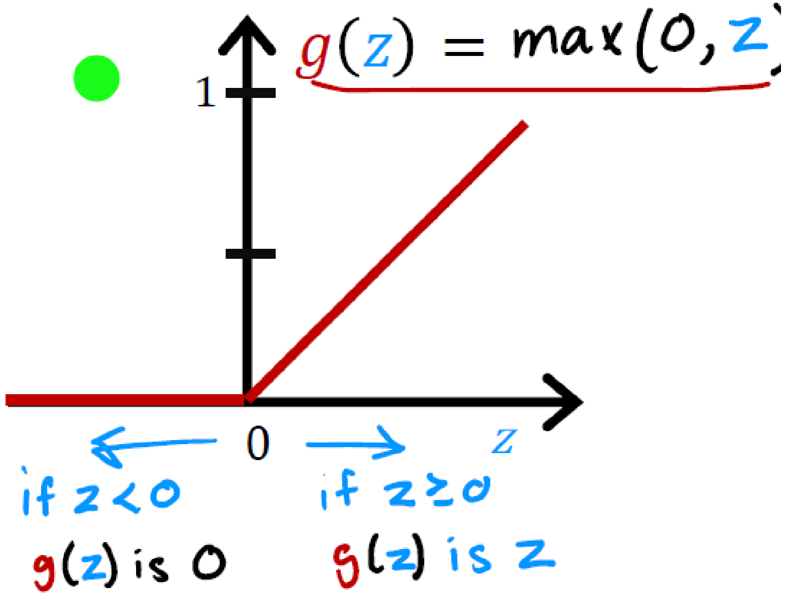
Sigmoid



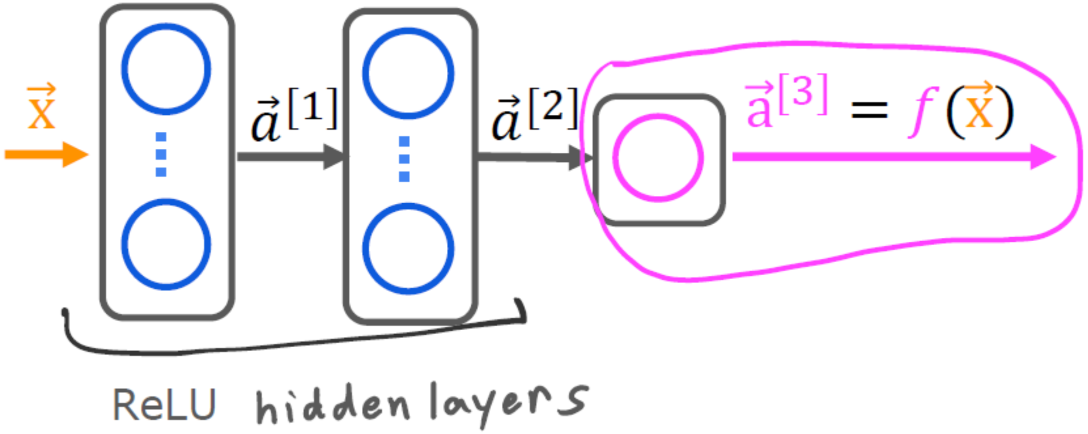
$0 < g(z) < 1$

Later: softmax activation

ReLU Rectified Linear Unit



Activation Functions




```
from tf.keras.layers import Dense
model = Sequential([
  Dense(units=25, activation='relu'),    layer1
  Dense(units=15, activation='relu'),    layer2
  Dense(units=1,  activation='sigmoid')  layer3
])
```

or 'linear'
or 'relu'

binary classification

activation='sigmoid'

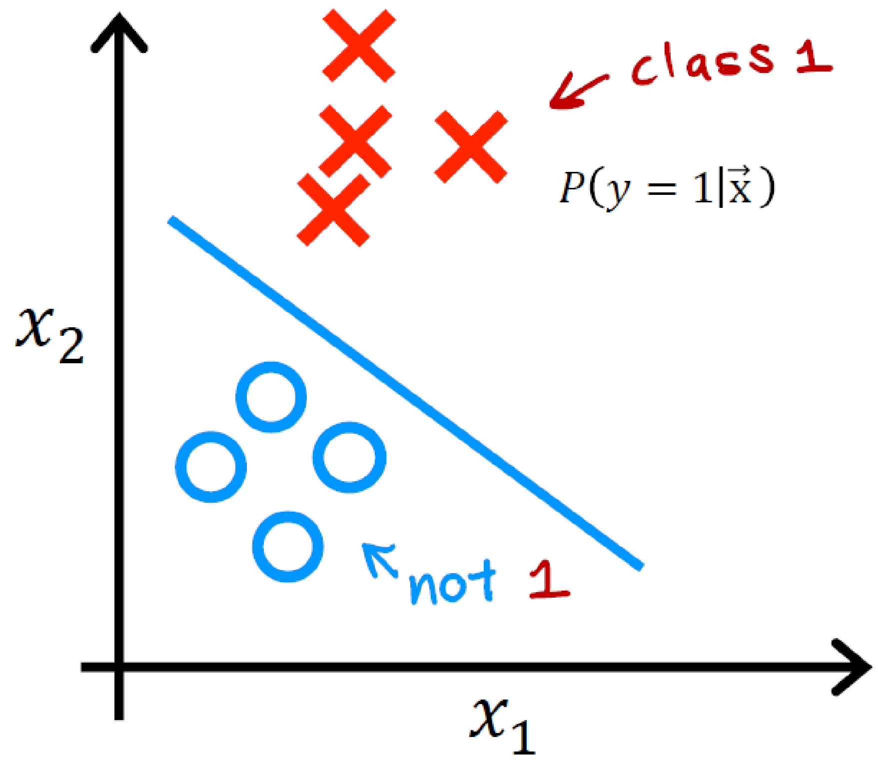
regression y negative/positive

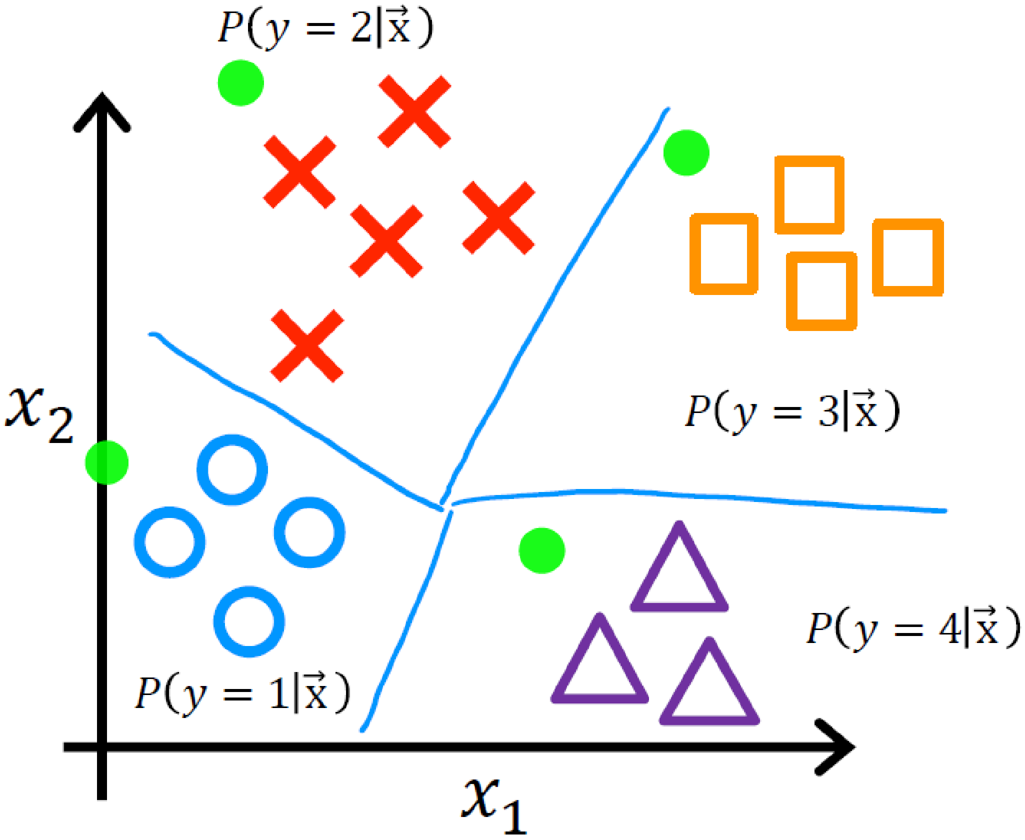
activation='linear'

regression $y \geq 0$

activation='relu'

Multiclass Classification





Multiclass Classification

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

Multiclass Classification

Softmax regression

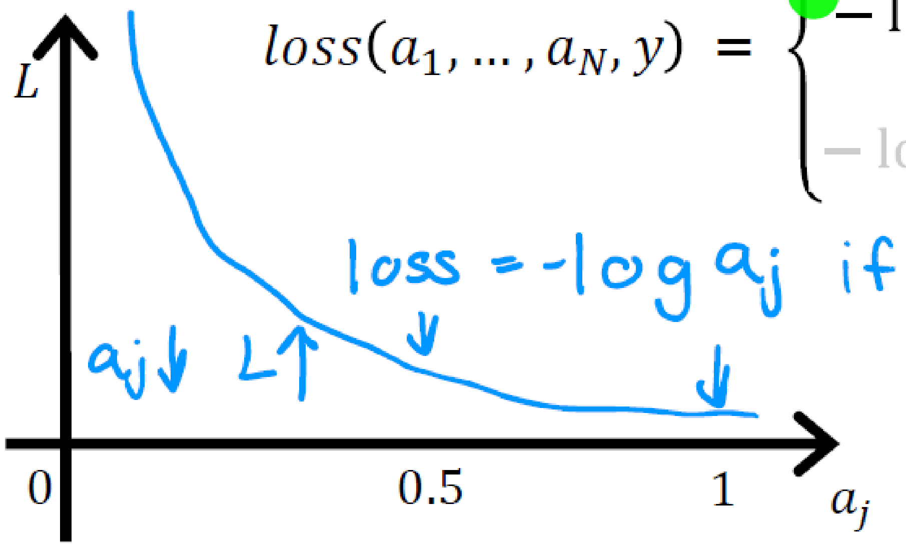
$$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + \cdots + e^{z_N}} = P(y = 1|\vec{\mathrm{x}})$$

$$\vdots$$

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

Crossentropy loss

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



Multiclass Classification

```
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)
```

THANK YOU

NEXT LECTURE WILL BE ONLINE ON SAT, 27.5.2023, IN SHAA ALLAH!

SABBAGH@IEEE.ORG

CONNECT ON LINKEDIN