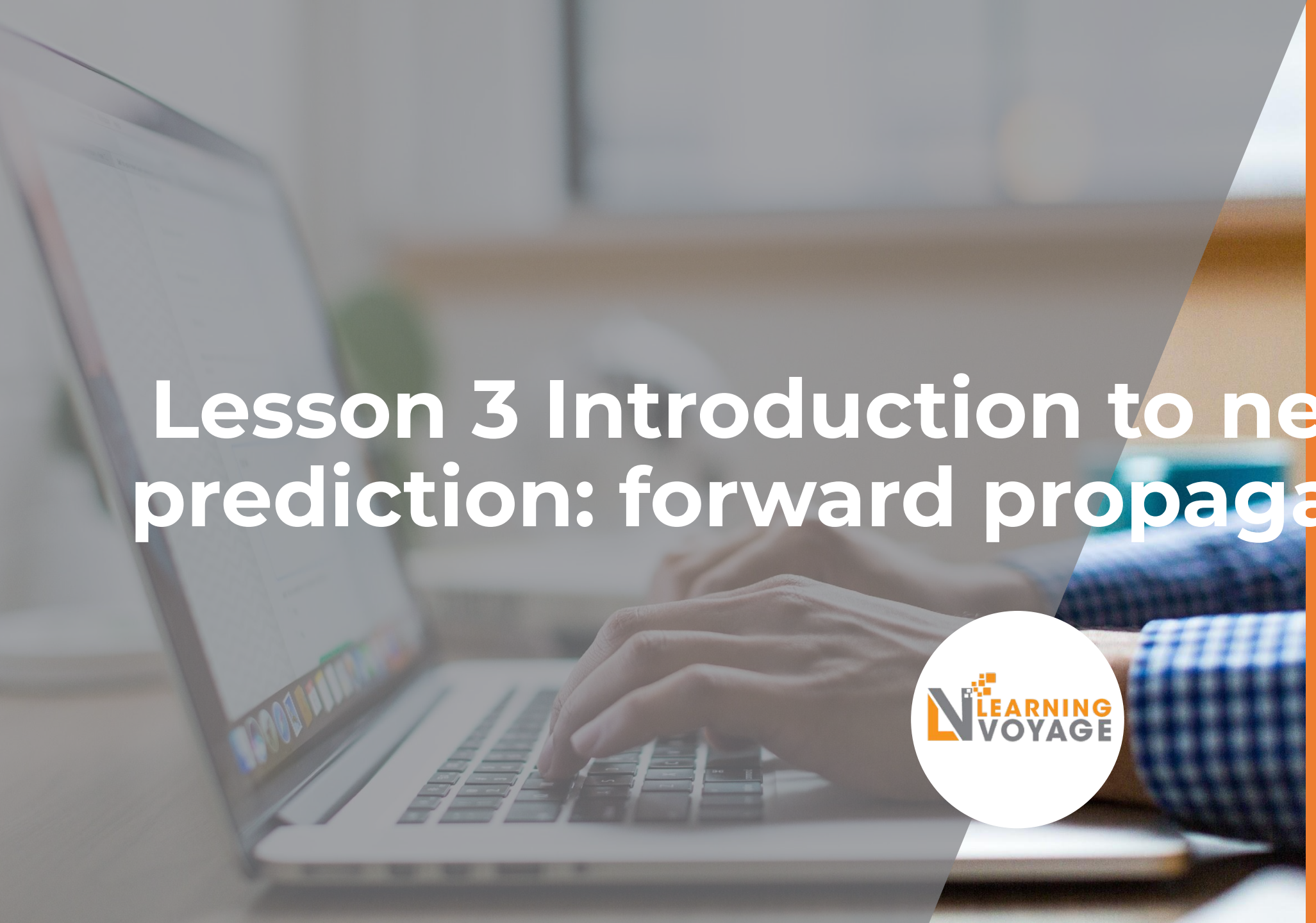


Lesson 3 Introduction to neural prediction: forward propagation

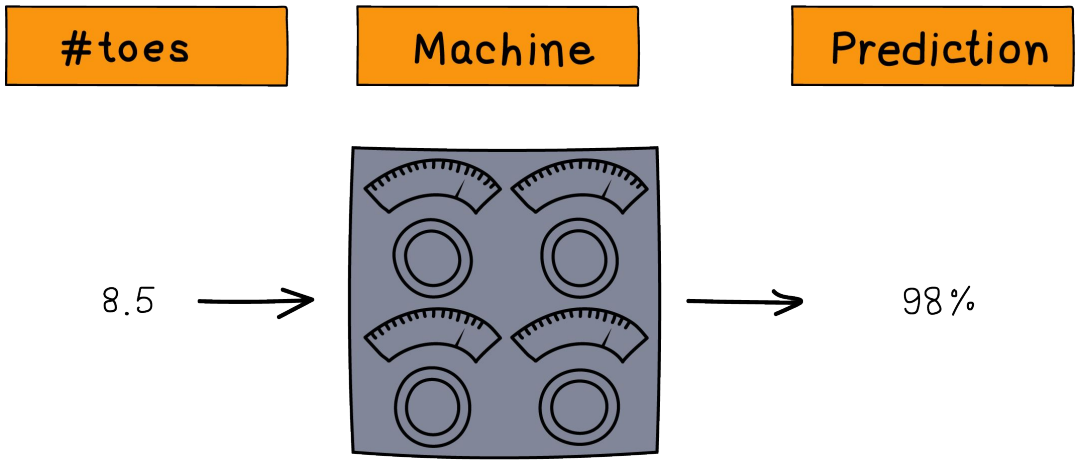
LEARNING VOYAGE

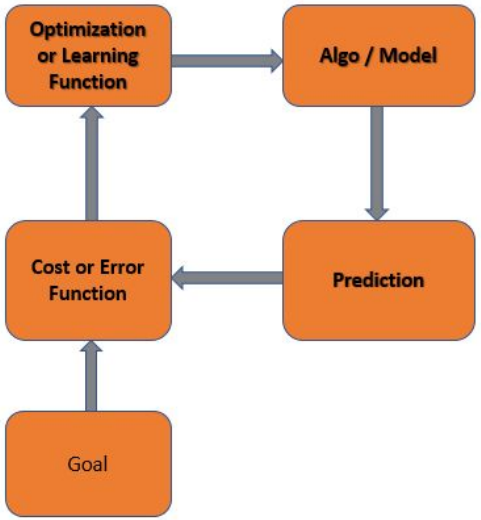
Forward propagation

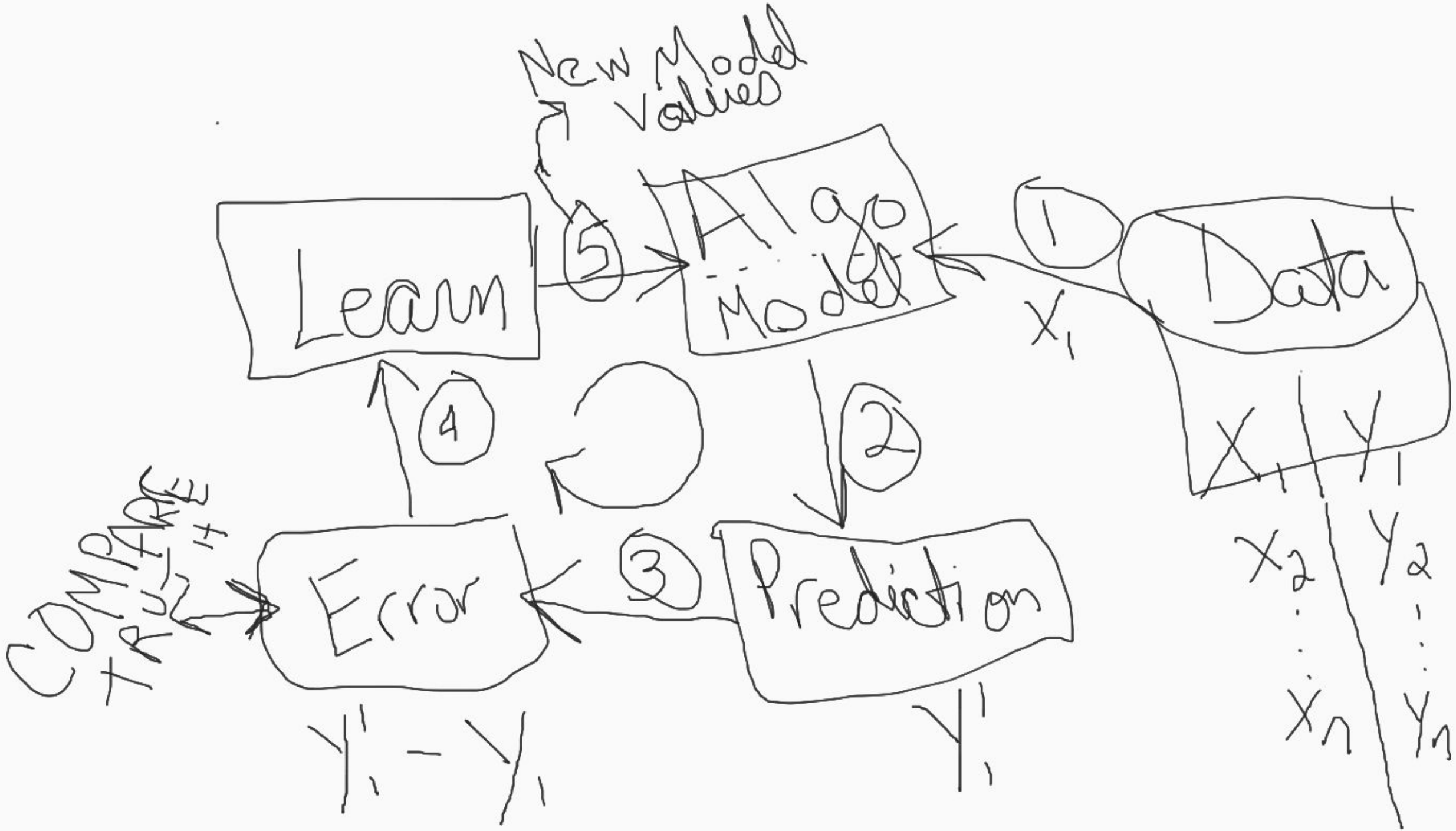
In this lesson we learn

- A simple network making a prediction
- What is a neural network, and what does it do?
- Making a prediction with multiple inputs
- Making a prediction with multiple outputs
- Making a prediction with multiple inputs and outputs
- Predicting on predictions

Forward propagation





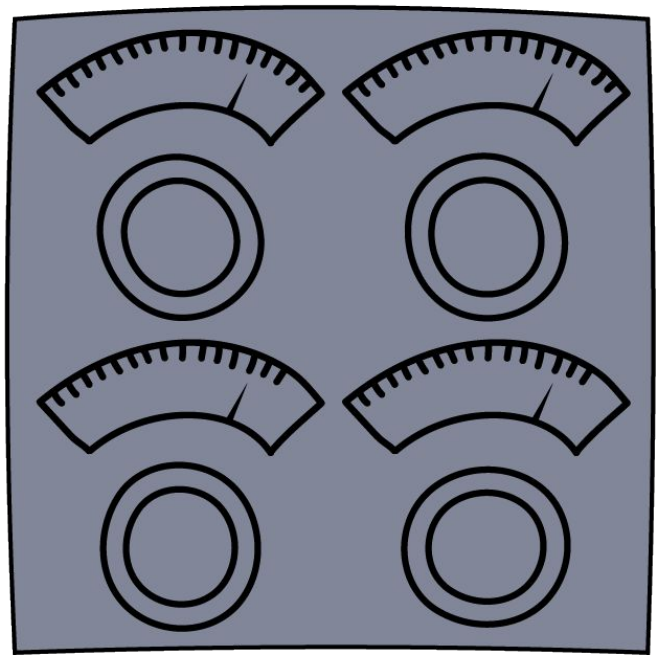

New Model Values
Learn
Algo
Model
Data
1
5
X_1
4
2
COMPARE TRUE
Error
3
Prediction
$Y'_1 - Y_1$
Y'_1
X_1 Y_1
X_2 Y_2
X_n Y_n

Forward propagation

- # Asking Price

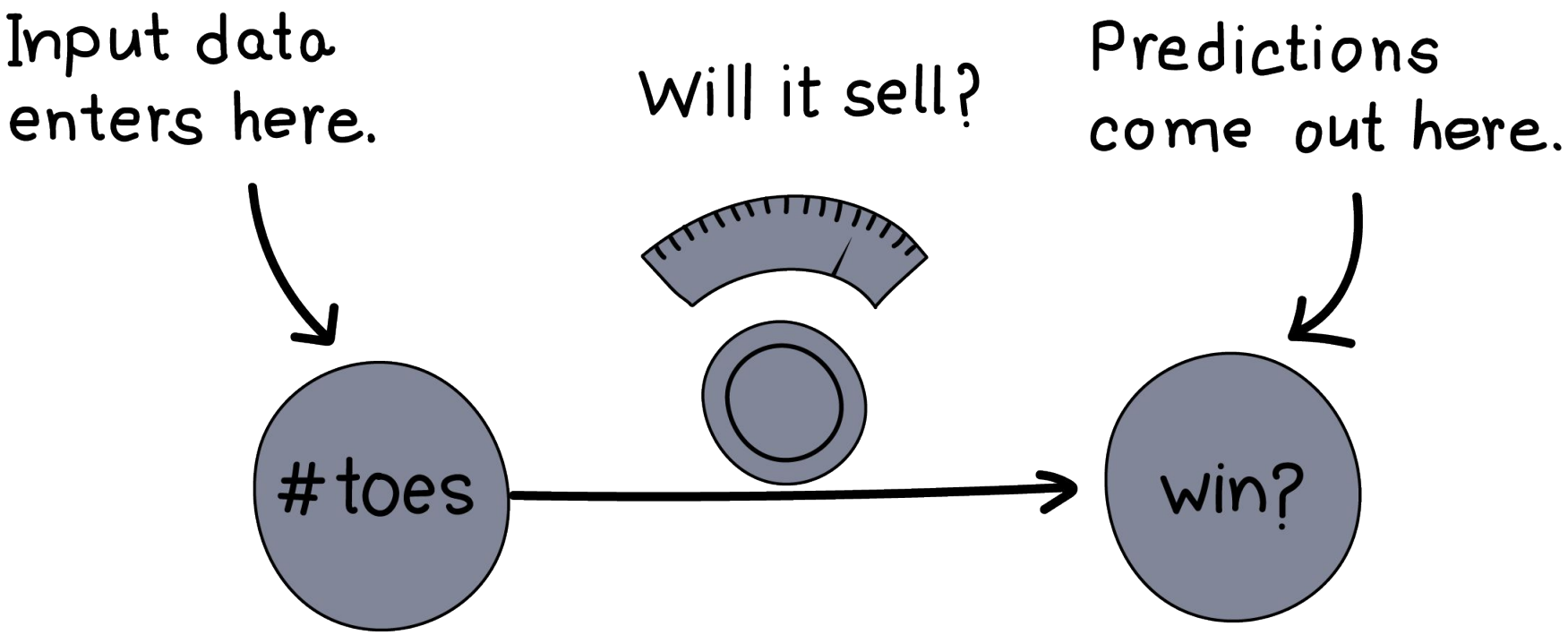
Machine

Selling probability

8.5 →  → 98 %

Forward propagation

① An empty network



A simple neural network making a prediction

What is a neural network?

```
# The network:
weight = 0.2
def neural_network(input, weight):
    prediction = input * weight
    return prediction
```

What is a neural network?

```
# How we use the network to predict something:
number_of_bedrooms = [3, 2, 4, 3, 5]
input = number_of_bedrooms[0]
pred = neural_network(input,weight)
print(pred)
```

What does this neural network do?

-

What does this neural network do?

-

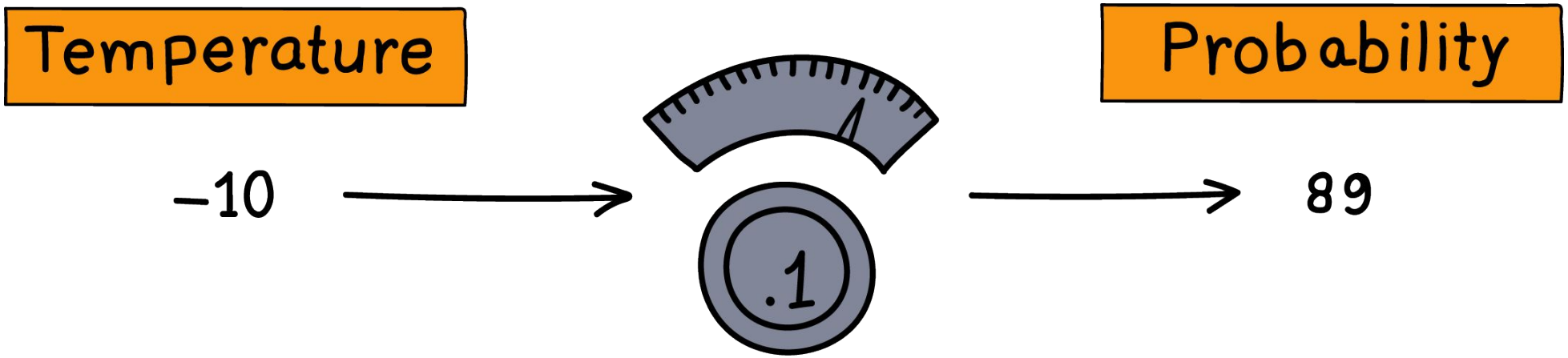
What does this neural network do?

-

What does this neural network do?

What does this neural network do?

What does this neural network do?



Making a prediction with multiple inputs

-

Making a prediction with multiple inputs

- Take a look at the next prediction:

Making a prediction with multiple inputs

Making a prediction with multiple inputs

Multiple inputs: What does this neural network do?

-

Multiple inputs: What does this neural network

-

Multiple inputs: What does this neural network do?

-

Multiple inputs: What does this neural network do?

Multiple inputs: What does this neural network do?

-

Multiple inputs: What does this neural network do?

-

```
a = [ 0,  1,  0,  1]
b = [ 1,  0,  1,  0]
c = [ 0,  1,  1,  0]
d = [.5,  0, .5,  0]
e = [ 0,  1, -1,  0]
```

```
w_sum(a,b)  = 0
w_sum(b,c)  = 1
w_sum(b,d)  = 1
w_sum(c,c)  = 2
w_sum(d,d)  = .5
w_sum(c,e)  = 0
```

Multiple inputs: What does this neural network do?

- These assume you're performing w_sum(input,weights) and the "then" to these if statements is an abstract "then give high score":

```
weights = [ 1,  0,  1] => if input[0] OR input[2]
weights = [ 0,  0,  1] => if input[2]
weights = [ 1,  0, -1] => if input[0] OR NOT input[2]
weights = [-1,  0, -1] => if NOT input[0] OR NOT input[2]
weights = [ 0.5, 0, 1] => if BIG input[0] or input[2]
```

Multiple inputs: What does this neural network do?

Multiple inputs: Complete runnable code

- The code snippets from this example come together in the following code, which creates and executes a neural network.

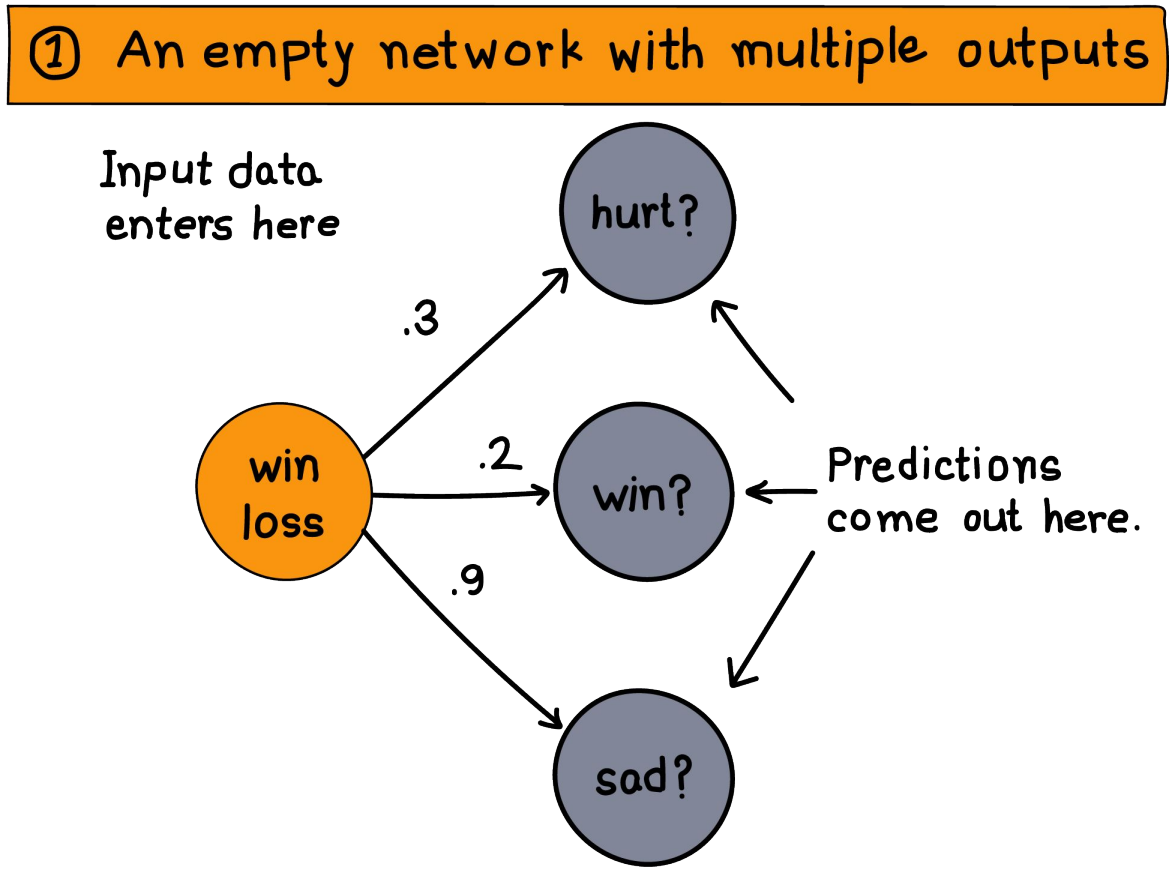
```
weights = [0.2, 0.01, 0.2, -0.2]

def neural_network(input, weights):
    pred = w_sum(input,weights)
    return pred

bedrooms =  [3, 2, 4, 3, 5]
bathrooms = [1, 1, 3, 2, 4]
floors = [1, 1, 2, 2, 1]
condition = [0.48, 0.85, 0.56, 0.2, 0.1]

# Input corresponds to the first house
input = [bedrooms[0], bathrooms[0], floors[0], condition[0]]
pred = neural_network(input,weights)
print(pred)
```

Multiple inputs: Complete runnable code

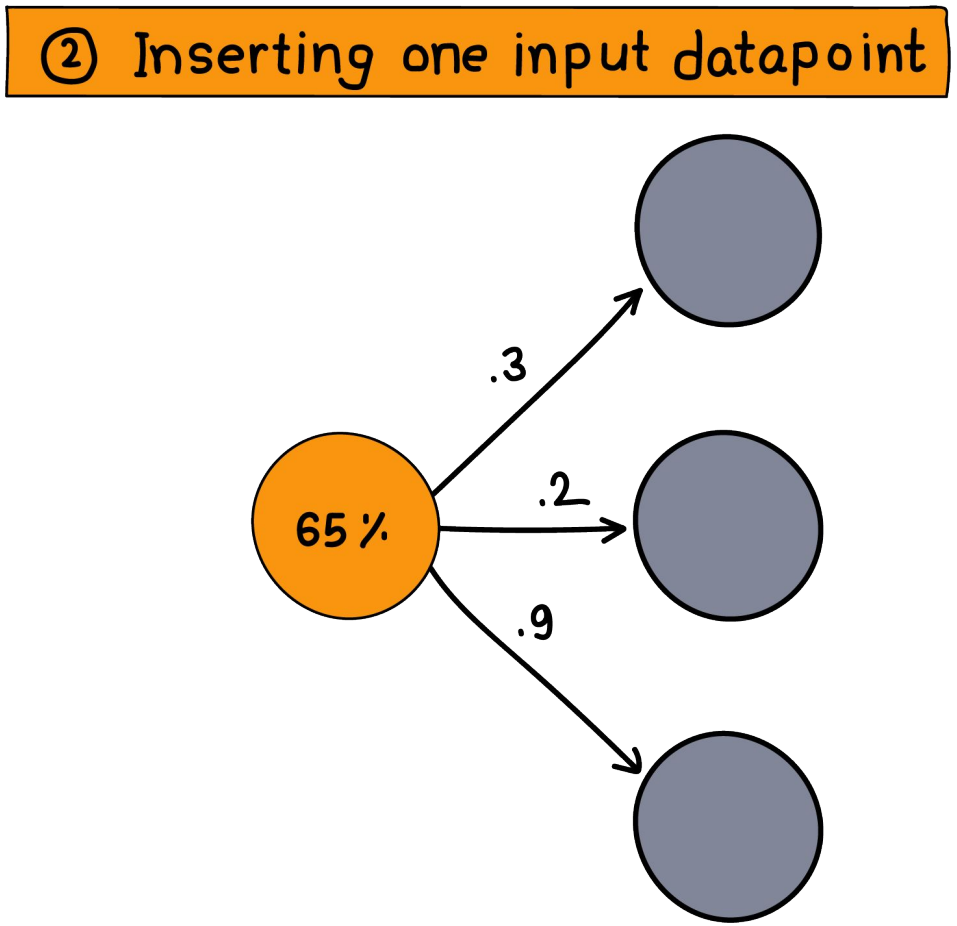
```
import numpy as np
weights = np.array([0.2, 0.01, 0.2, -0.2])
def neural_network(input, weights):
    pred = input.dot(weights)
    return pred

bedrooms =  np.array([3, 2, 4, 3, 5])
bathrooms = np.array([1, 1, 3, 2, 4])
floors = np.array([1, 1, 2, 2, 1])
condition = np.array([0.48, 0.85, 0.56, 0.2, 0.1])

# Input corresponds to the first house in the dataset.
input = np.array([bedrooms[0], bathrooms[0], floors[0], condition[0]])
pred = neural_network(input,weights)
print(pred)
```

Making a prediction with multiple outputs

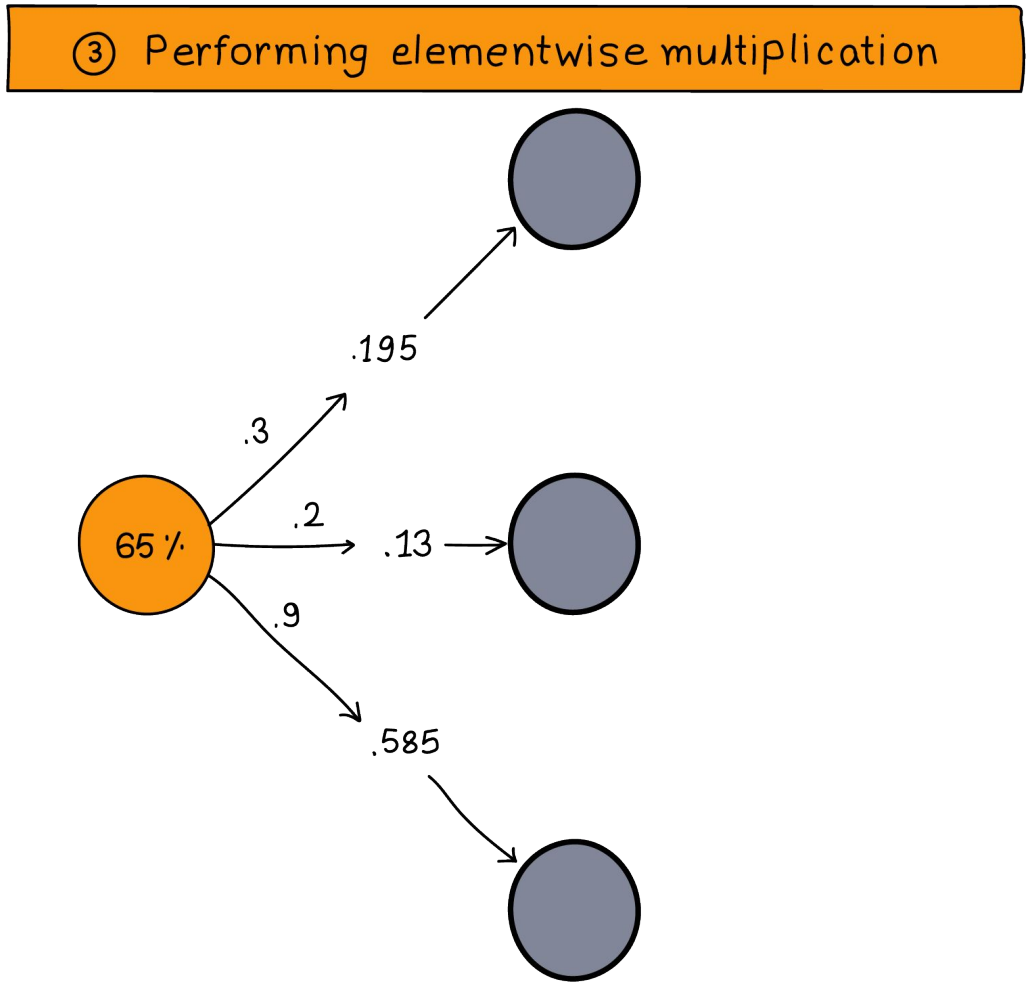
Neural networks can also make multiple predictions using only a single input.



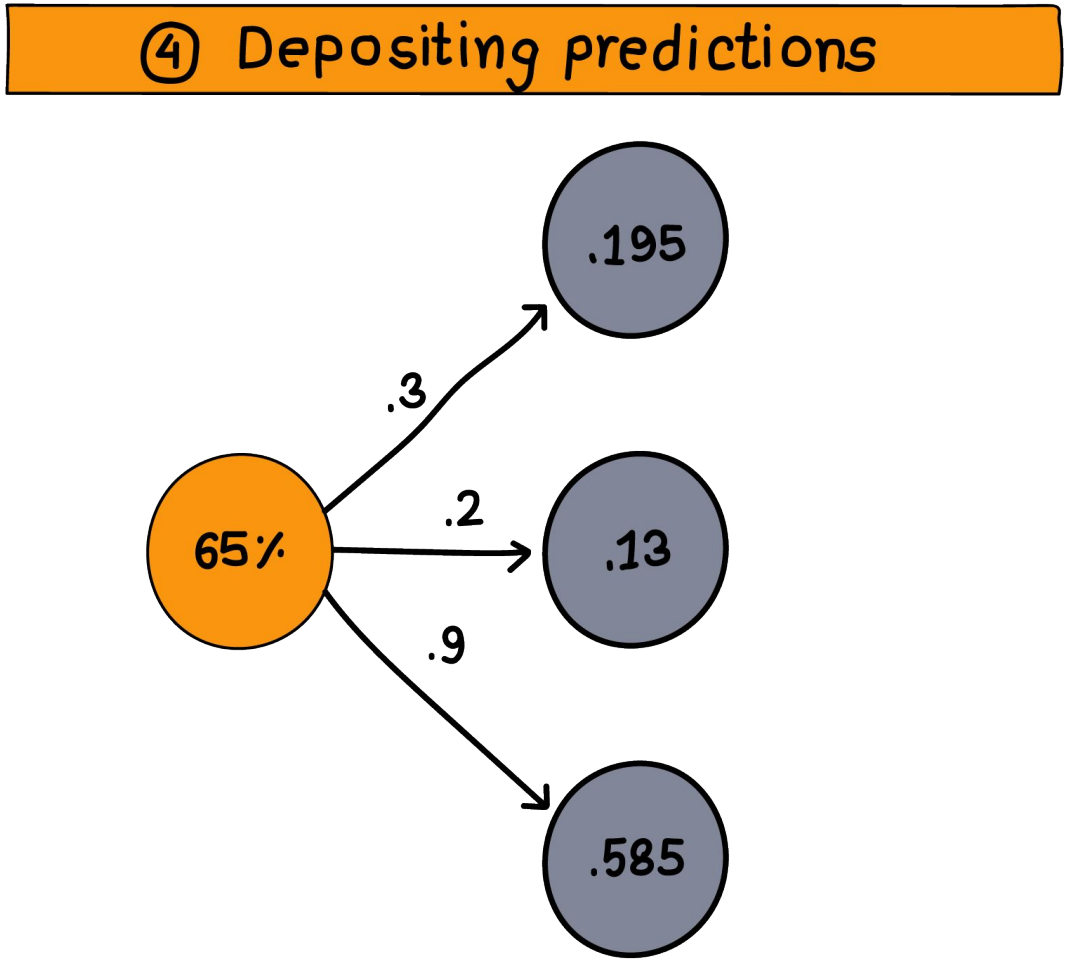
Making a prediction with multiple outputs



Making a prediction with multiple outputs

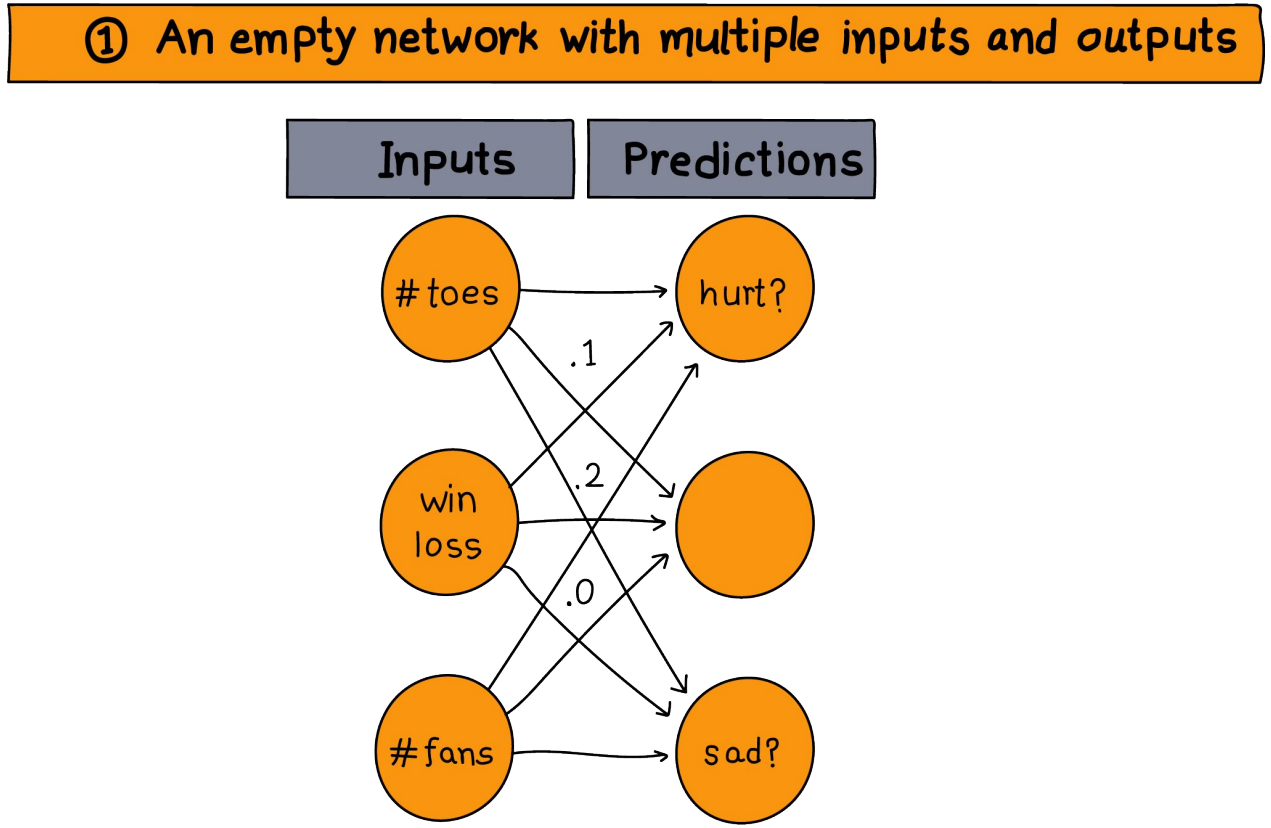


Making a prediction with multiple outputs

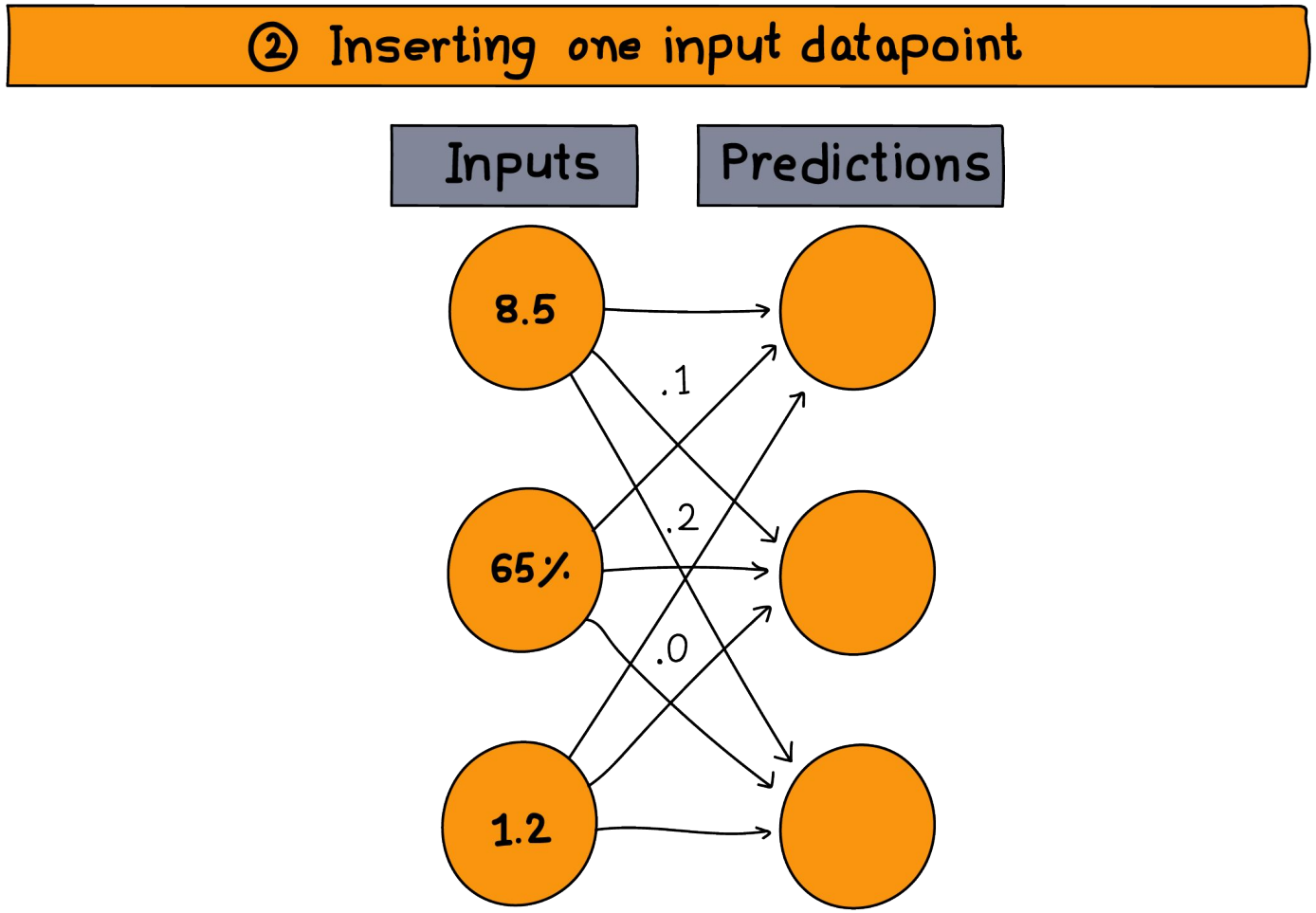


Predicting with multiple inputs and outputs

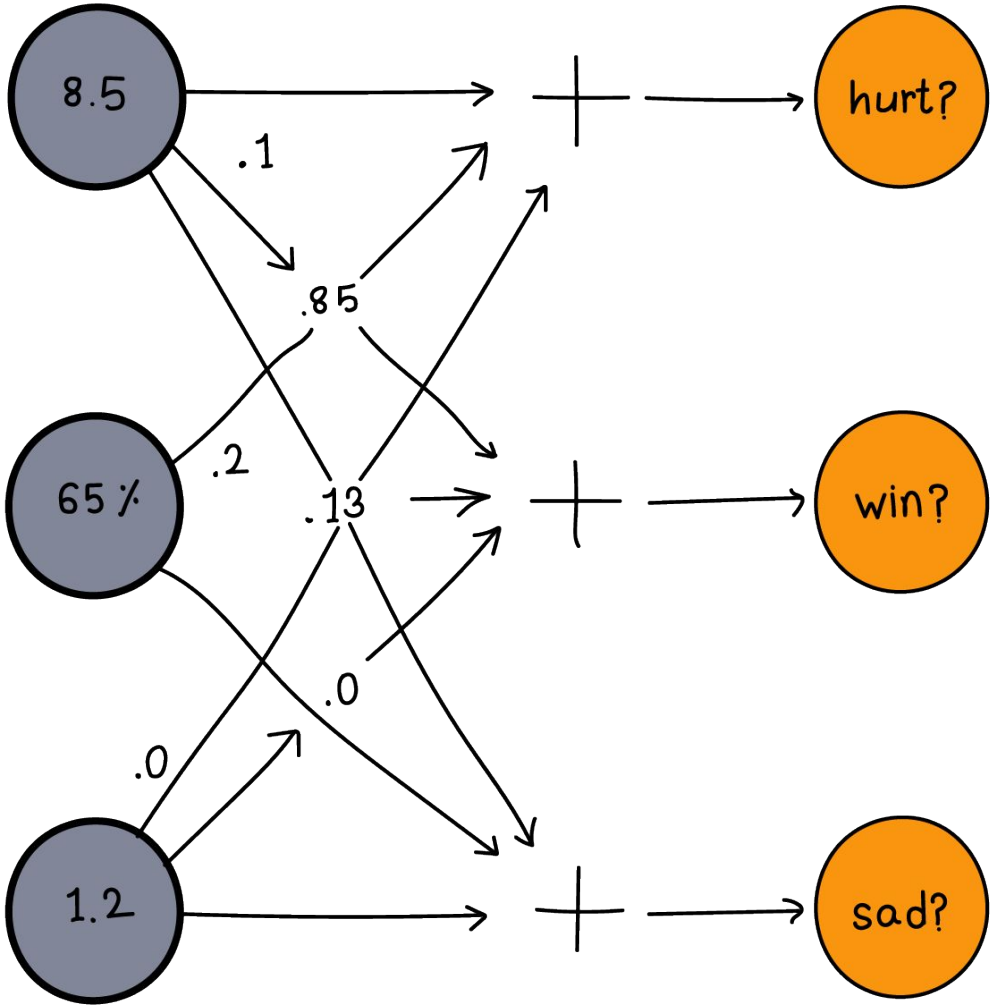
Neural networks can predict multiple outputs given multiple inputs



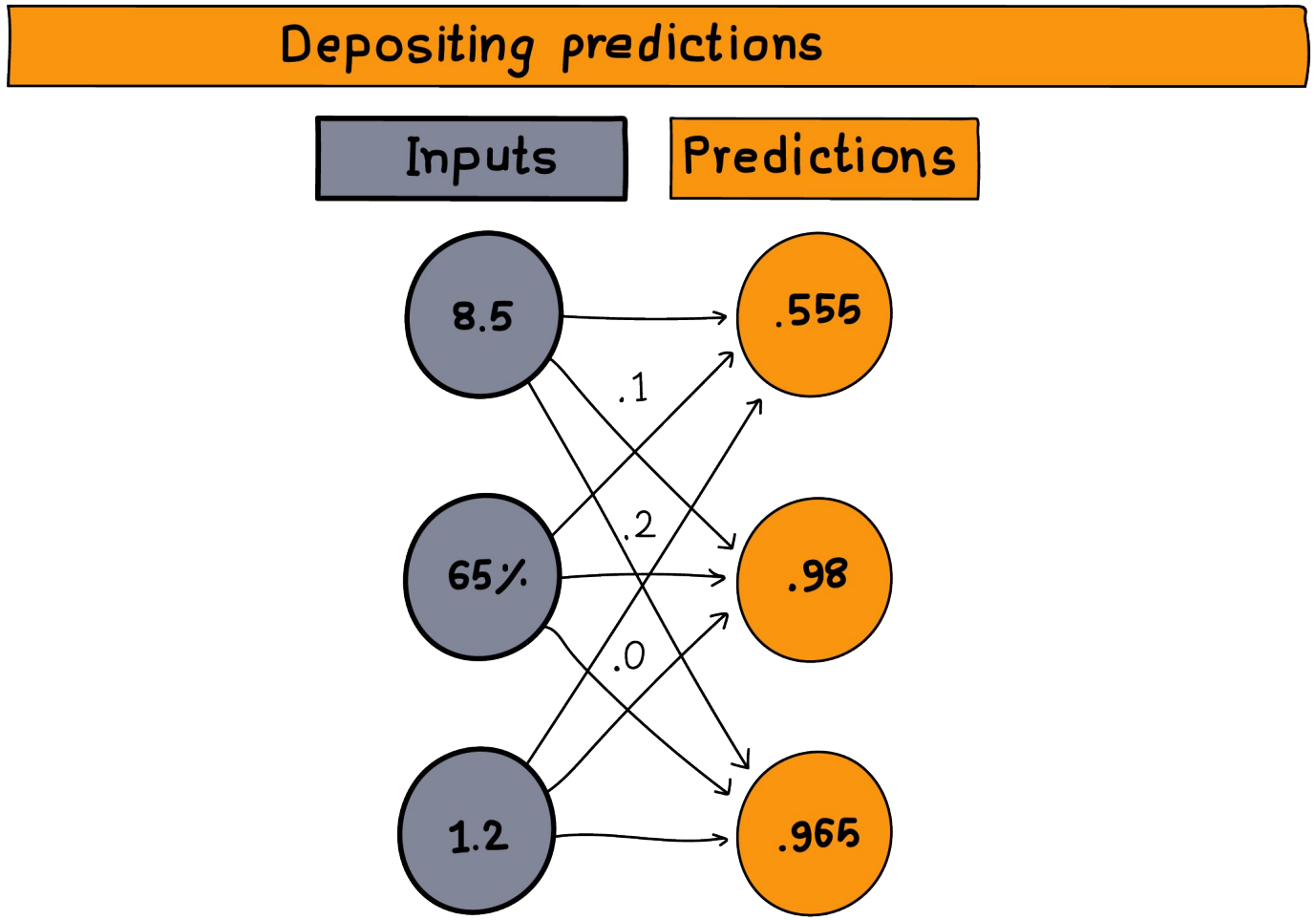
Predicting with multiple inputs and outputs



③ For each output, performing a weighted sum of inputs

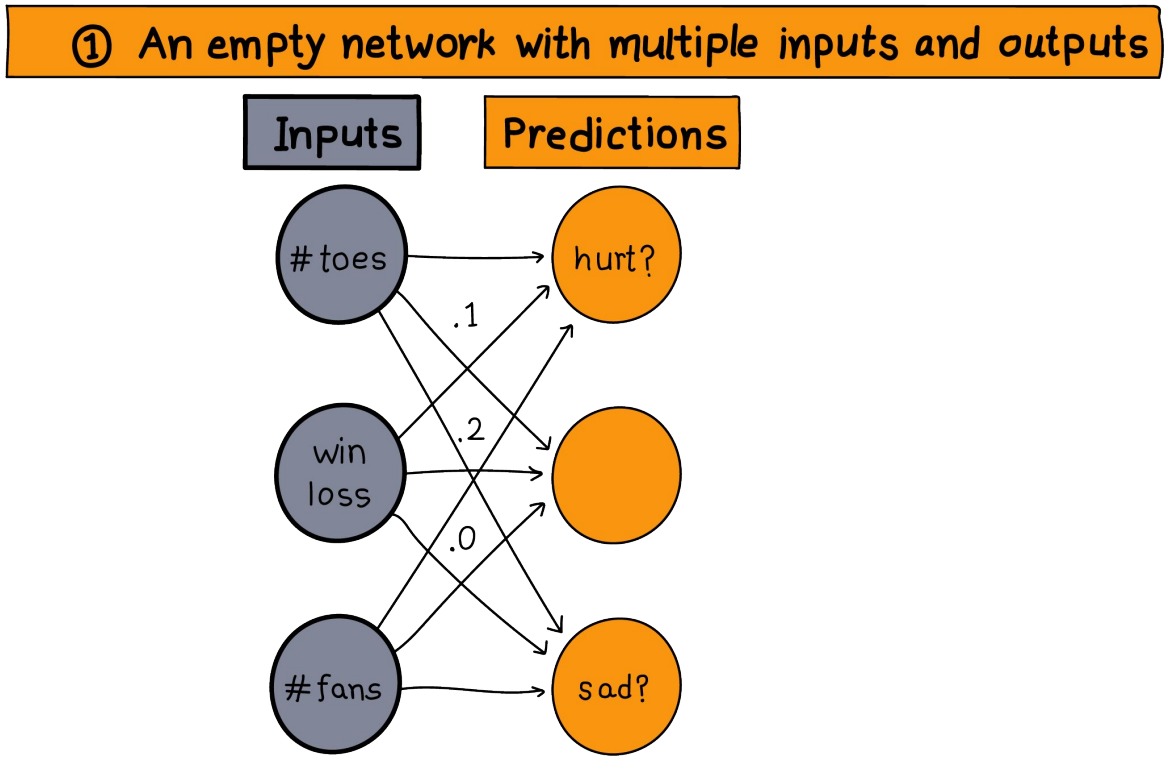


Predicting with multiple inputs and outputs

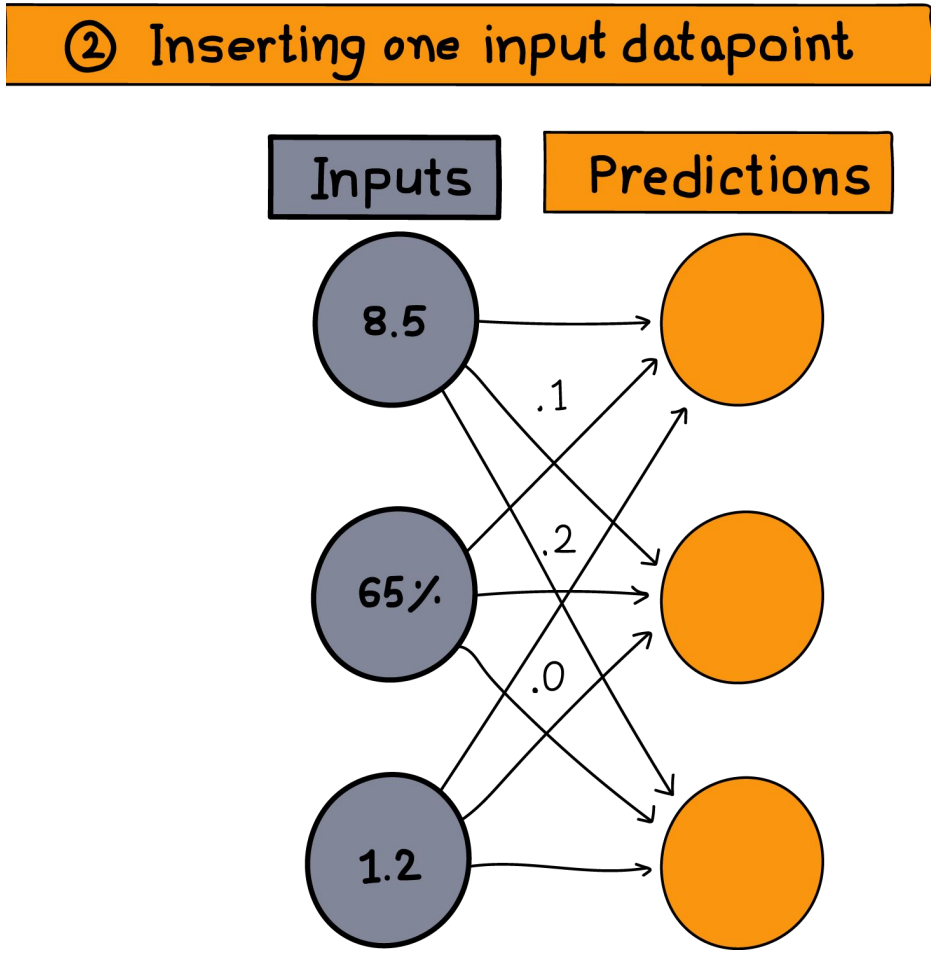


Multiple inputs and outputs: How does it work?

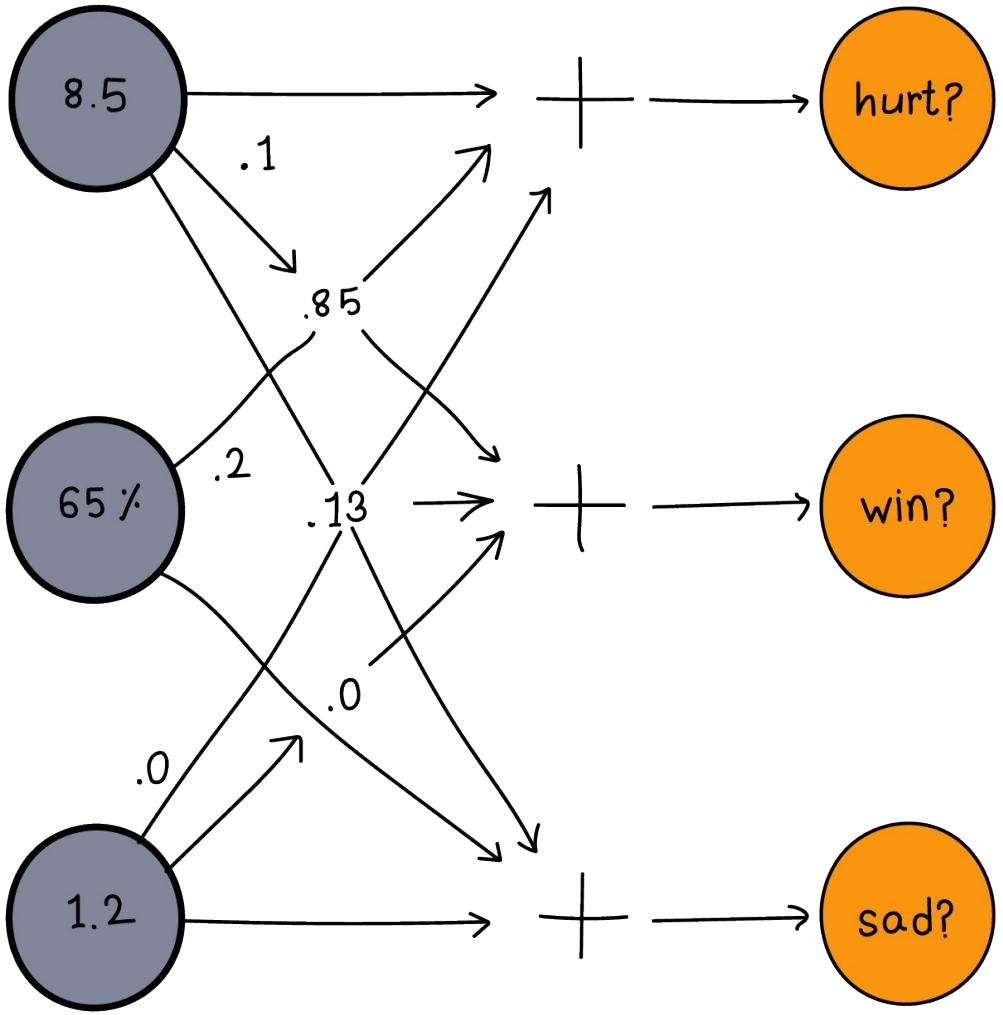
It performs three independent weighted sums of the input to make three predictions.



Multiple inputs and outputs: How does it work?



③ For each output, performing a weighted sum of inputs



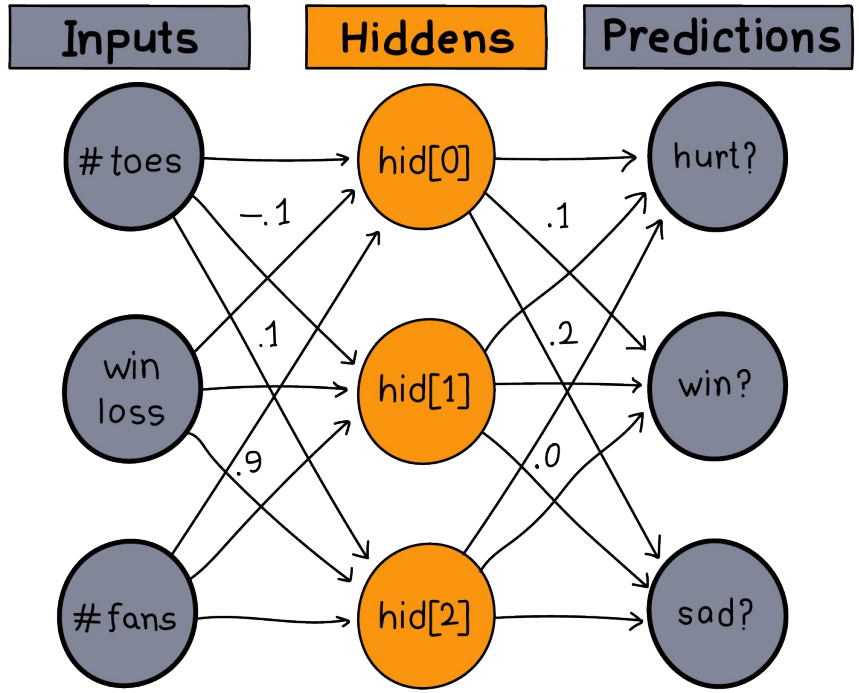
Multiple inputs and outputs: How does it work?

-

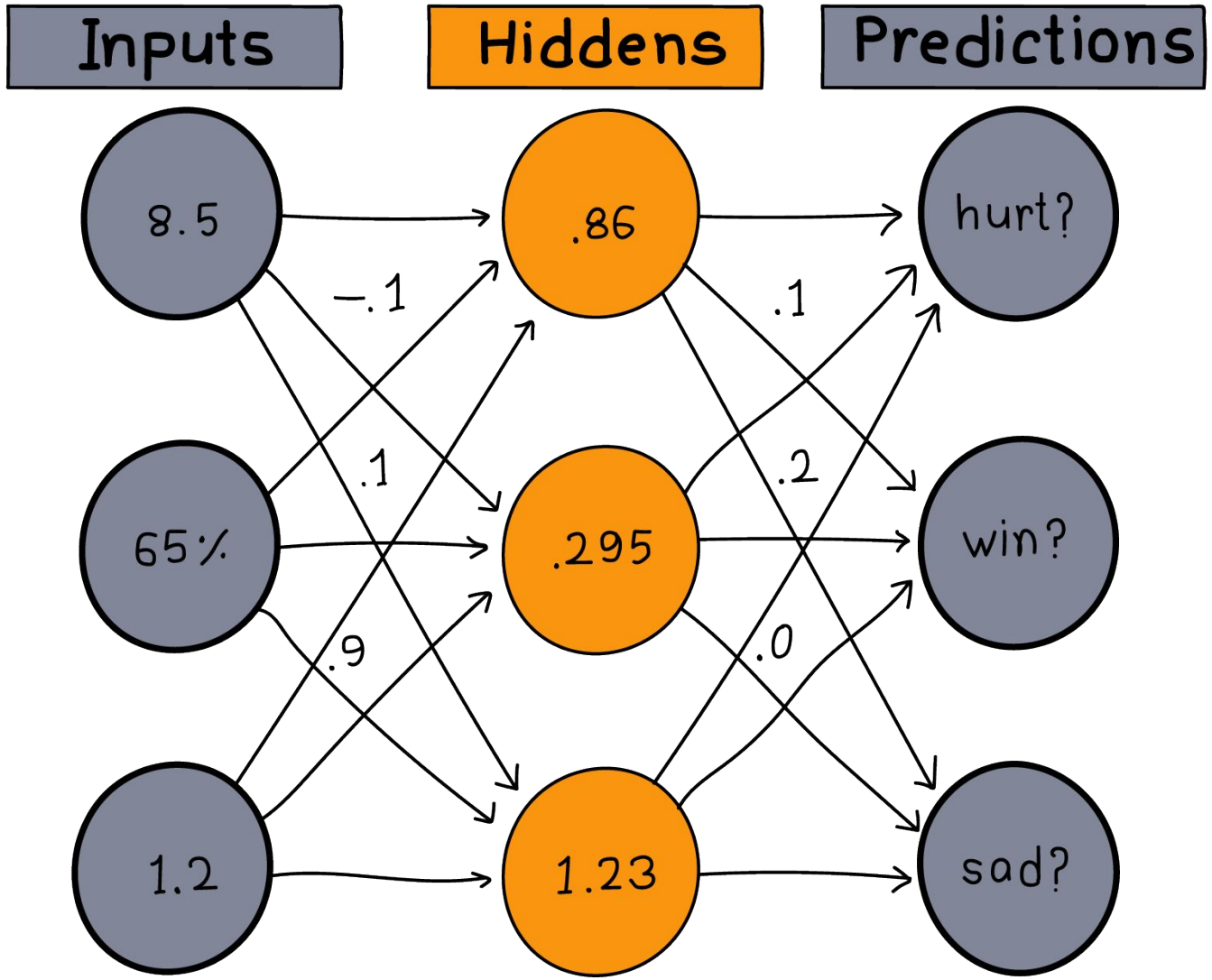
Predicting on predictions

Neural networks can be stacked!

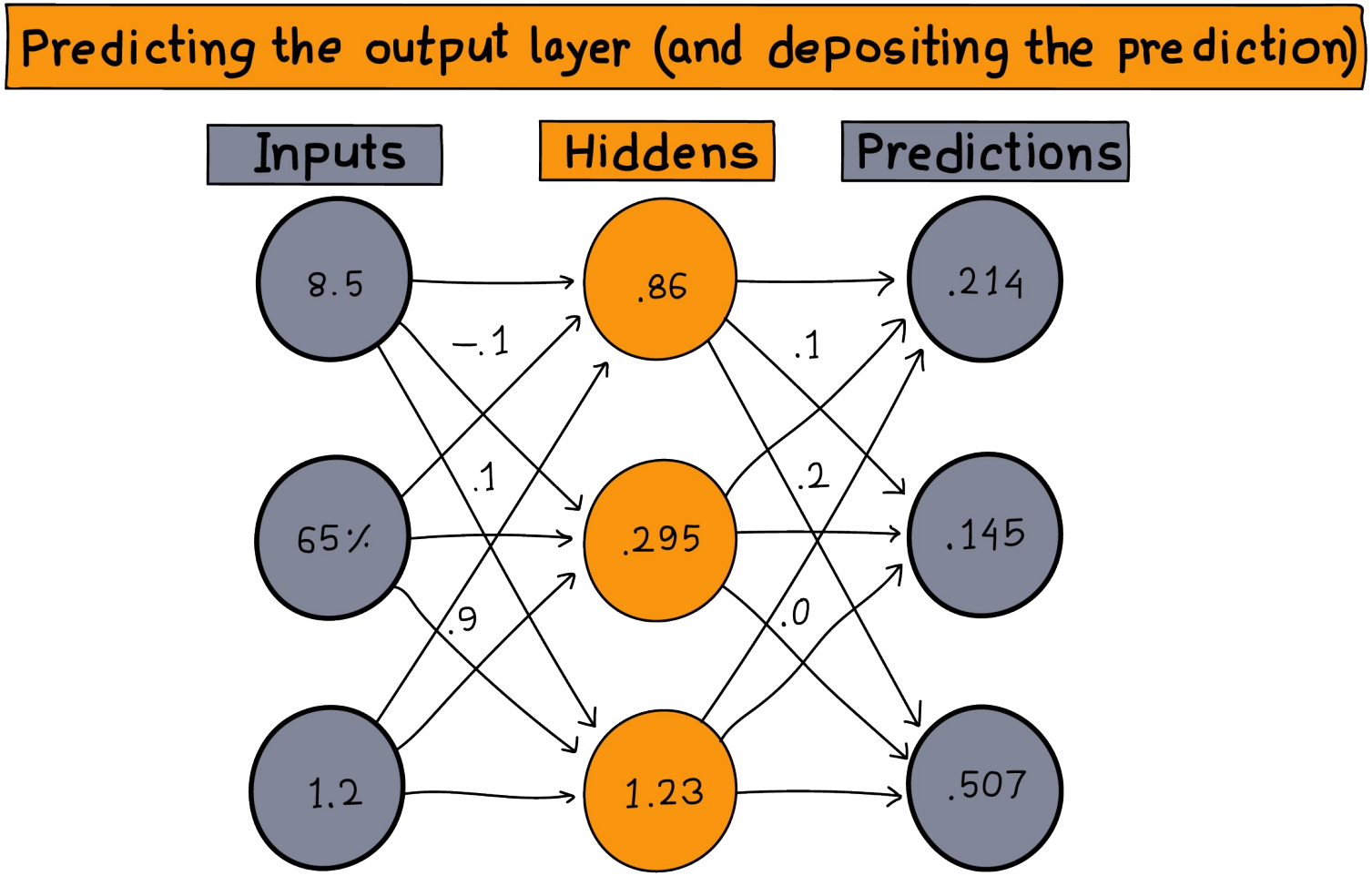
① An empty network with multiple inputs and outputs



② Predicting the hidden layer



Predicting on predictions



```
import numpy as np

# toes % win # fans
ih_wgt = np.array([
             [0.1, 0.2, -0.1], # hid[0]
             [-0.1,0.1, 0.9], # hid[1]
             [0.1, 0.4, 0.1]]).T # hid[2]


# hid[0] hid[1] hid[2]
hp_wgt = np.array([
             [0.3, 1.1, -0.3], # hurt?
             [0.1, 0.2, 0.0], # win?
             [0.0, 1.3, 0.1] ]).T # sad?

weights = [ih_wgt, hp_wgt]

def neural_network(input, weights):

    hid = input.dot(weights[0])
    pred = hid.dot(weights[1])
    return pred


toes =  np.array([8.5, 9.5, 9.9, 9.0])
wlrec = np.array([0.65,0.8, 0.8, 0.9])
nfans = np.array([1.2, 1.3, 0.5, 1.0])

input = np.array([toes[0],wlrec[0],nfans[0]])

pred = neural_network(input,weights)
print(pred)
```

NumPy version

A quick primer on NumPy

NumPy does a few things for you. Let's reveal the magic.

- You've also learned about different operations that occur on vectors and matrices, including dot products, elementwise multiplication and addition, and vector-matrix multiplication.

- For these operations, you've written Python functions that can operate on simple Python list objects.

A quick primer on NumPy

- You can create vectors and matrices in multiple ways in NumPy.
- Most of the common techniques for neural networks are listed in the previous code.

A quick primer on NumPy

```
import numpy as np

a = np.array([0,1,2,3])
b = np.array([4,5,6,7])
c = np.array([[0,1,2,3],
              [4,5,6,7]])

d = np.zeros((2,4))
e = np.random.rand(2,5)

print(a)
print(b)
print(c)
print(d)
print(e)
```

A vector

Another vector

A matrix

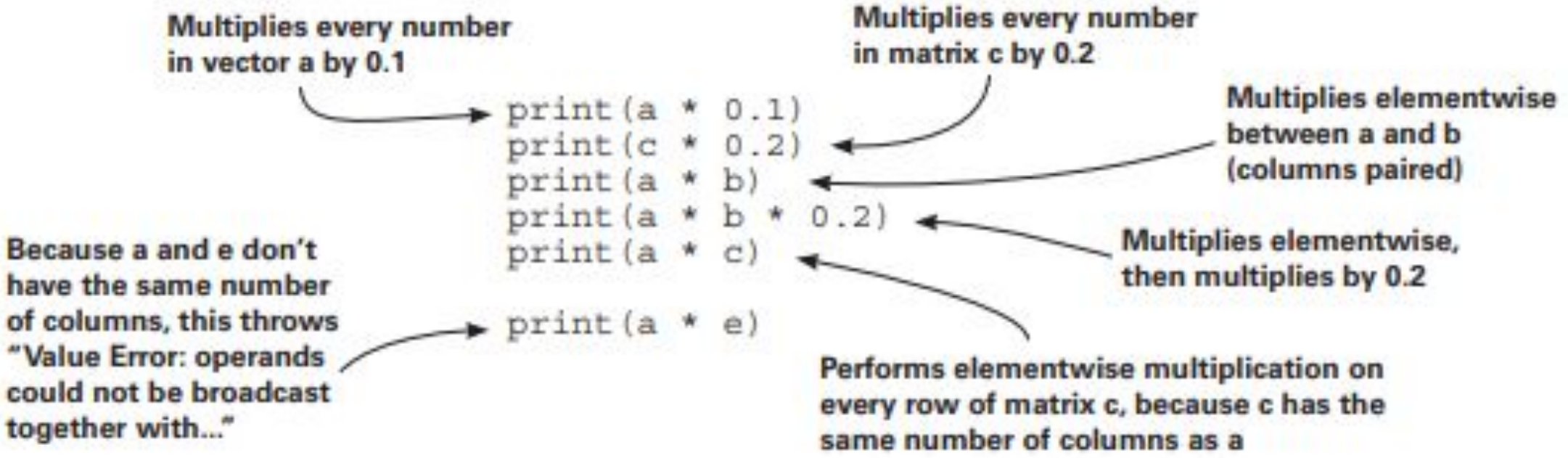
2 x 4 matrix of zeros

Random 2 x 5 matrix of numbers between 0 and 1

Output

```
[0 1 2 3]
[4 5 6 7]
[[0 1 2 3]
 [4 5 6 7]]
[[ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]]
[[ 0.22717119  0.39712632
0.0627734   0.08431724
0.53469141]
 [ 0.09675954  0.99012254
0.45922775  0.3273326
0.28617742]]
```

A quick primer on NumPy



A quick primer on NumPy

- The general rule of thumb for anything elementwise (+, –, *, /) is that either the two variables must have the same number of columns, or one of the variables must have only one column.
- For example, print(a * 0.1) multiplies a vector by a single number (a scalar). NumPy says, “Oh, I bet I’m supposed to do vector-scalar multiplication here,” and then multiples the scalar (0.1) by every value in the vector.
- This looks exactly the same as print(c * 0.2), except NumPy knows that c is a matrix.

A quick primer on NumPy

- Again, the most confusing part is that all of these operations look the same if you don't know which variables are scalars, vectors, or matrices.

- When you "read NumPy," you're really doing two things: reading the operations and keeping track of the shape (number of rows and columns) of each operation.

A quick primer on NumPy

- Let's look at a few examples of matrix multiplication in NumPy, noting the input and output shapes of each matrix.

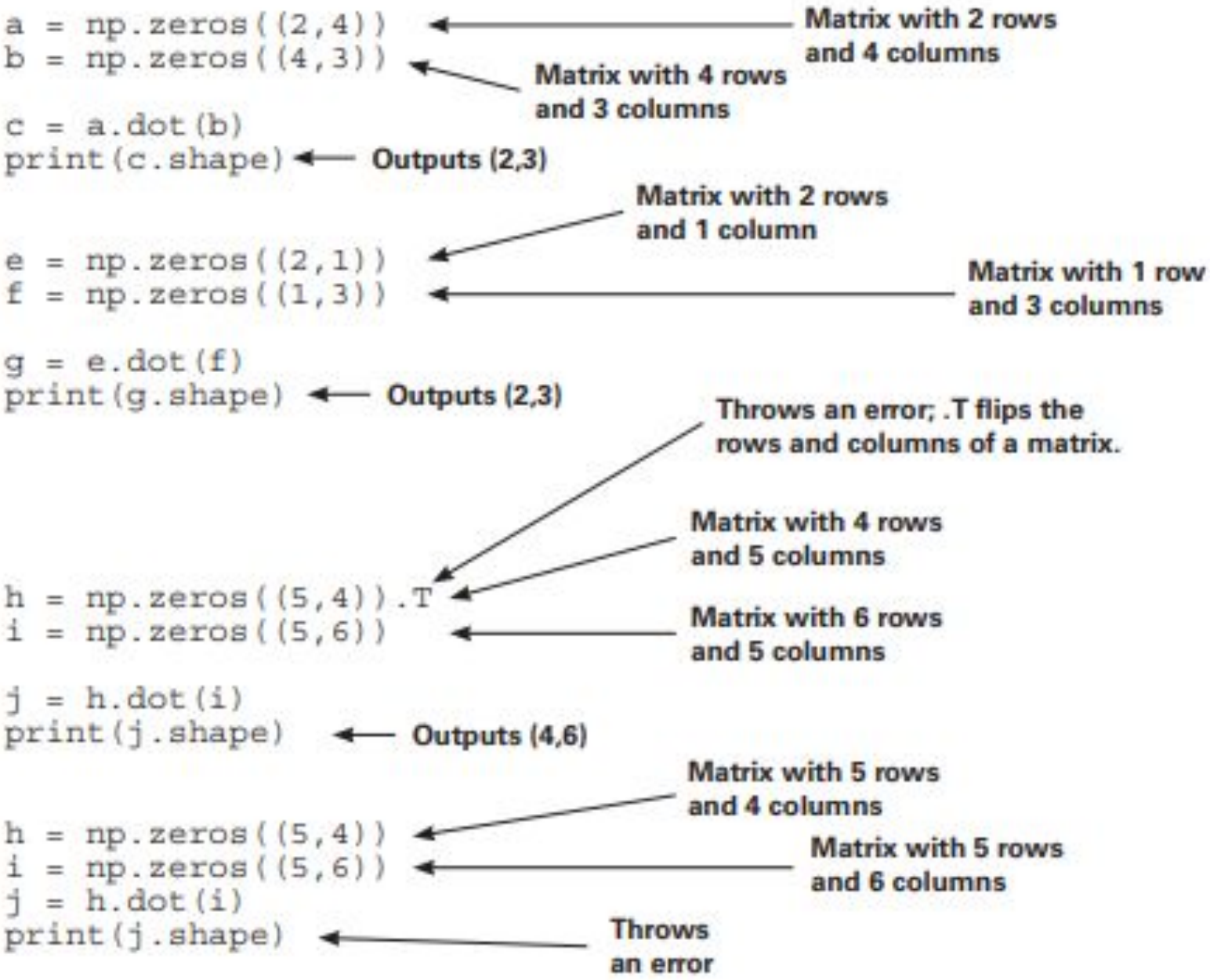
```
a = np.zeros((1,4))
b = np.zeros((4,3))

c = a.dot(b)
print(c.shape)
```

Vector of length 4

Matrix with 4 rows and 3 columns

Output

```
(1,3)
```


a = np.zeros((2,4))
b = np.zeros((4,3))
Matrix with 2 rows and 4 columns
Matrix with 4 rows and 3 columns
c = a.dot(b)
print(c.shape)
Outputs (2,3)
Matrix with 2 rows and 1 column
e = np.zeros((2,1))
f = np.zeros((1,3))
Matrix with 1 row and 3 columns
g = e.dot(f)
print(g.shape)
Outputs (2,3)
Throws an error; .T flips the rows and columns of a matrix.
Matrix with 4 rows and 5 columns
h = np.zeros((5,4)).T
i = np.zeros((5,6))
Matrix with 6 rows and 5 columns
j = h.dot(i)
print(j.shape)
Outputs (4,6)
Matrix with 5 rows and 4 columns
h = np.zeros((5,4))
i = np.zeros((5,6))
Matrix with 5 rows and 6 columns
j = h.dot(i)
print(j.shape)
Throws an error

Summary

To predict, neural networks perform repeated weighted sums of the input.

- You've seen an increasingly complex variety of neural networks in this lesson.
- I hope it's clear that a relatively small number of simple rules are used repeatedly to create larger, more advanced neural networks.
- The network's intelligence depends on the weight values you give it.