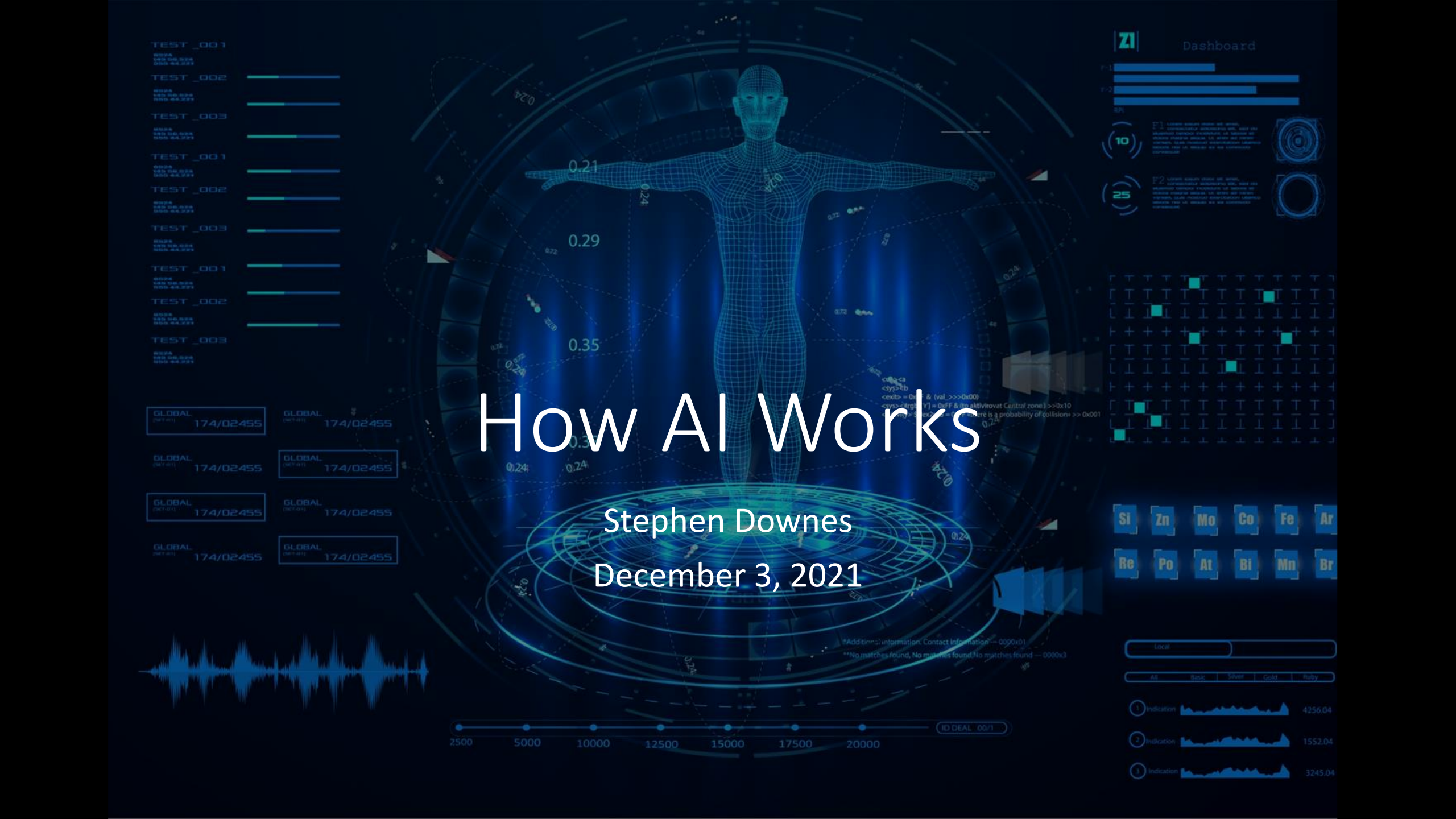


How AI Works

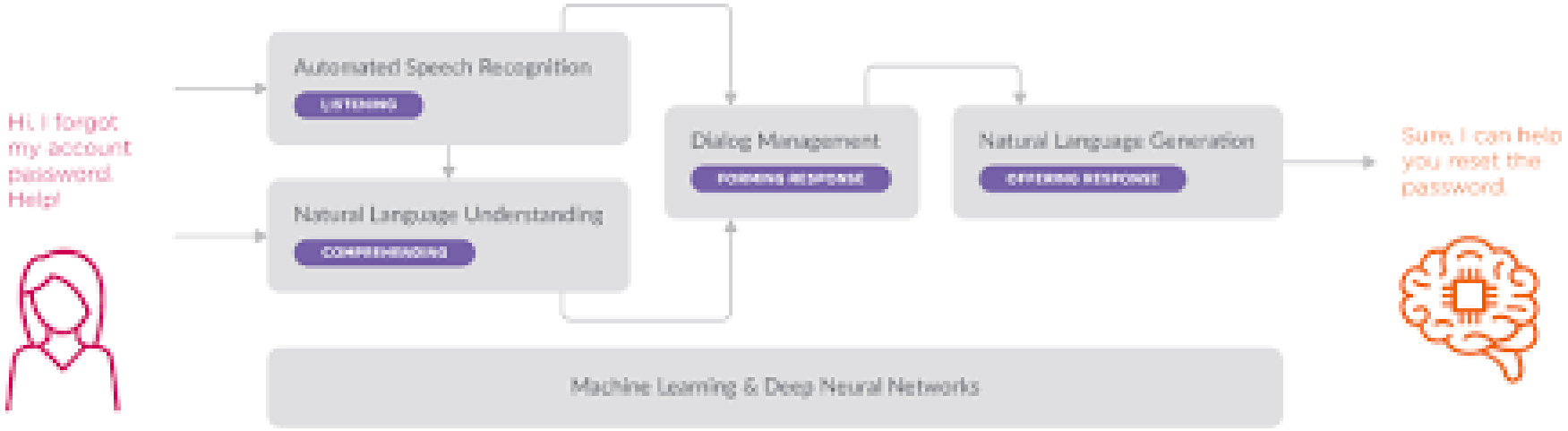
Stephen Downes

December 3, 2021

How AI Works

There are *many* good introductory videos for this subject:

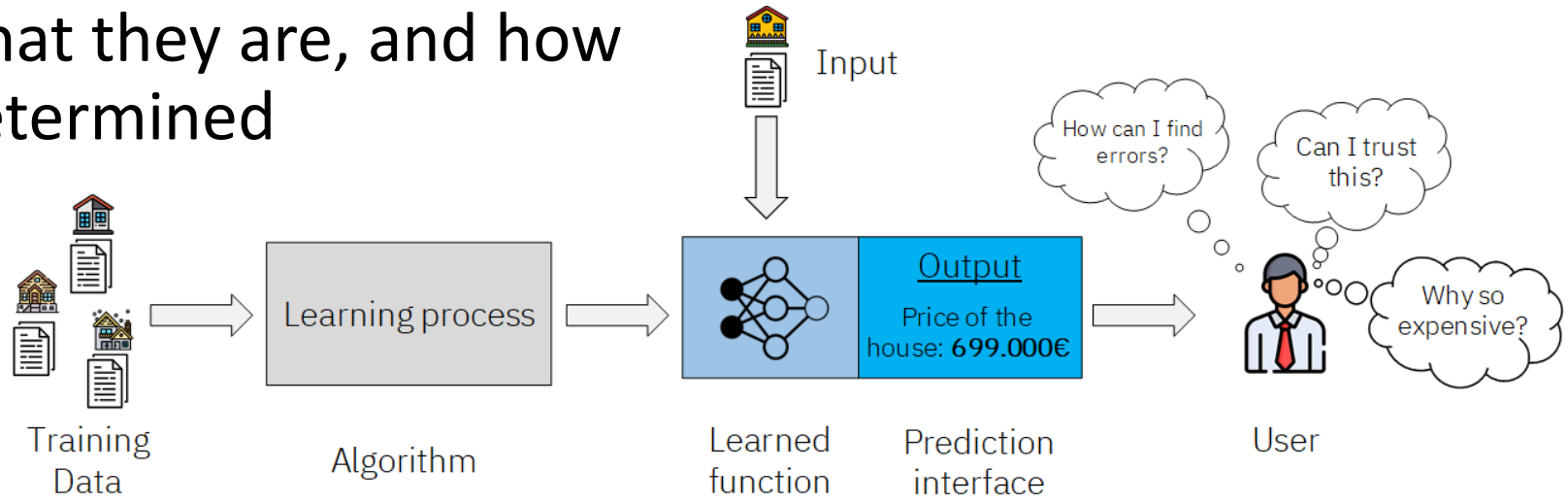
- https://www.youtube.com/watch?v=L_9OluD0nqw
- https://www.youtube.com/watch?v=JrXazCEACVo
- https://www.youtube.com/watch?v=aircAruvnKk



What Does AI Do?

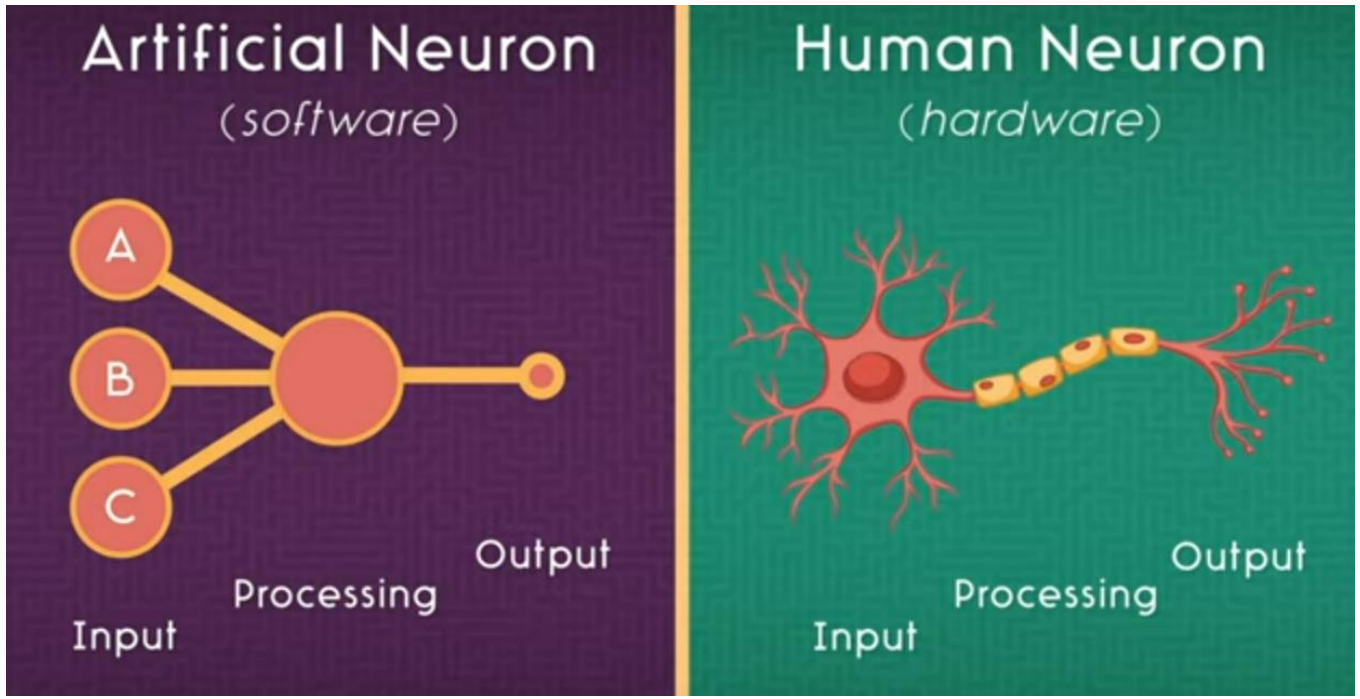
- A neural network is ultimately a statistical function, but one that manages tens of thousands of input variables
- This presentation is about those values: what they are, and how they're determined

- Regression
- Feature detection
- Clustering
- Prediction



Perceptrons

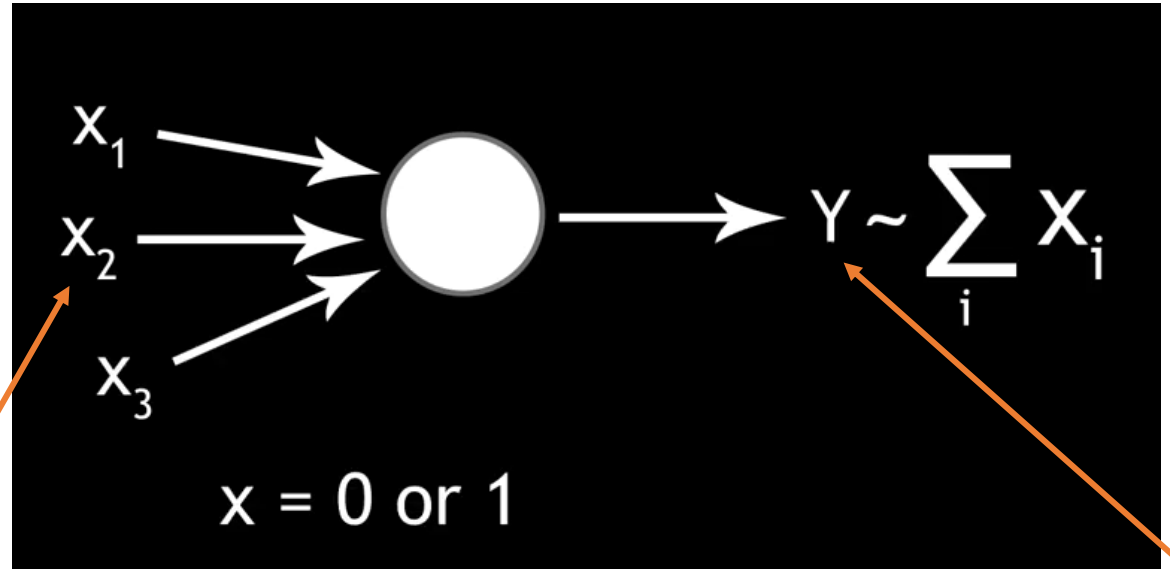
- A.k.a. an artificial neuron



https://www.youtube.com/watch?v=JrXazCEACVo

Perceptrons

- A.k.a. an artificial neuron



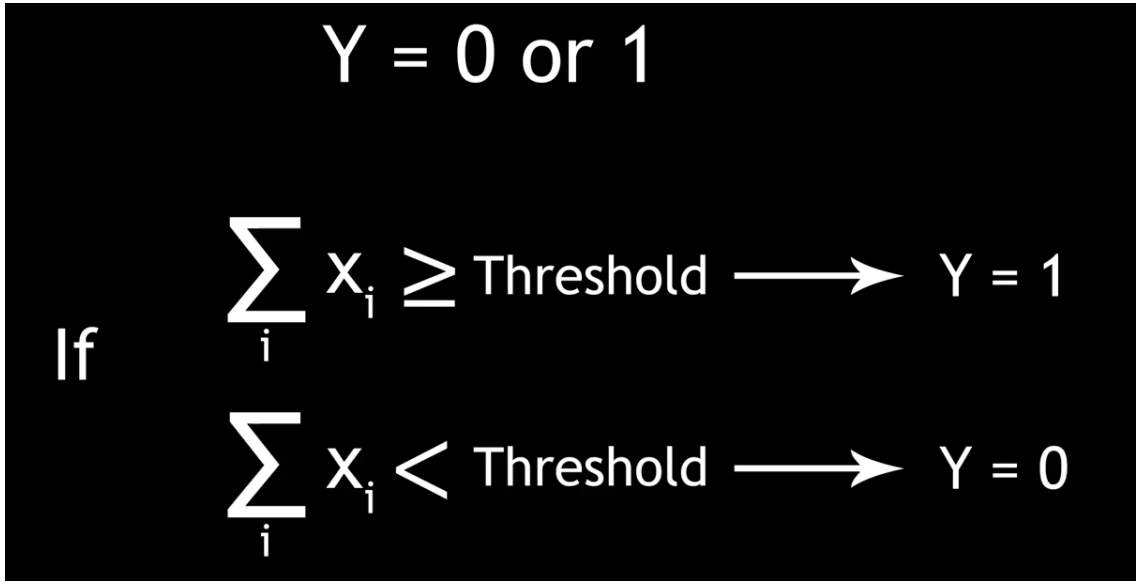
input values **x**

output value **y**, which depends on the sum of the input values

https://www.youtube.com/watch?v=L_9OluD0nqw

Threshholds

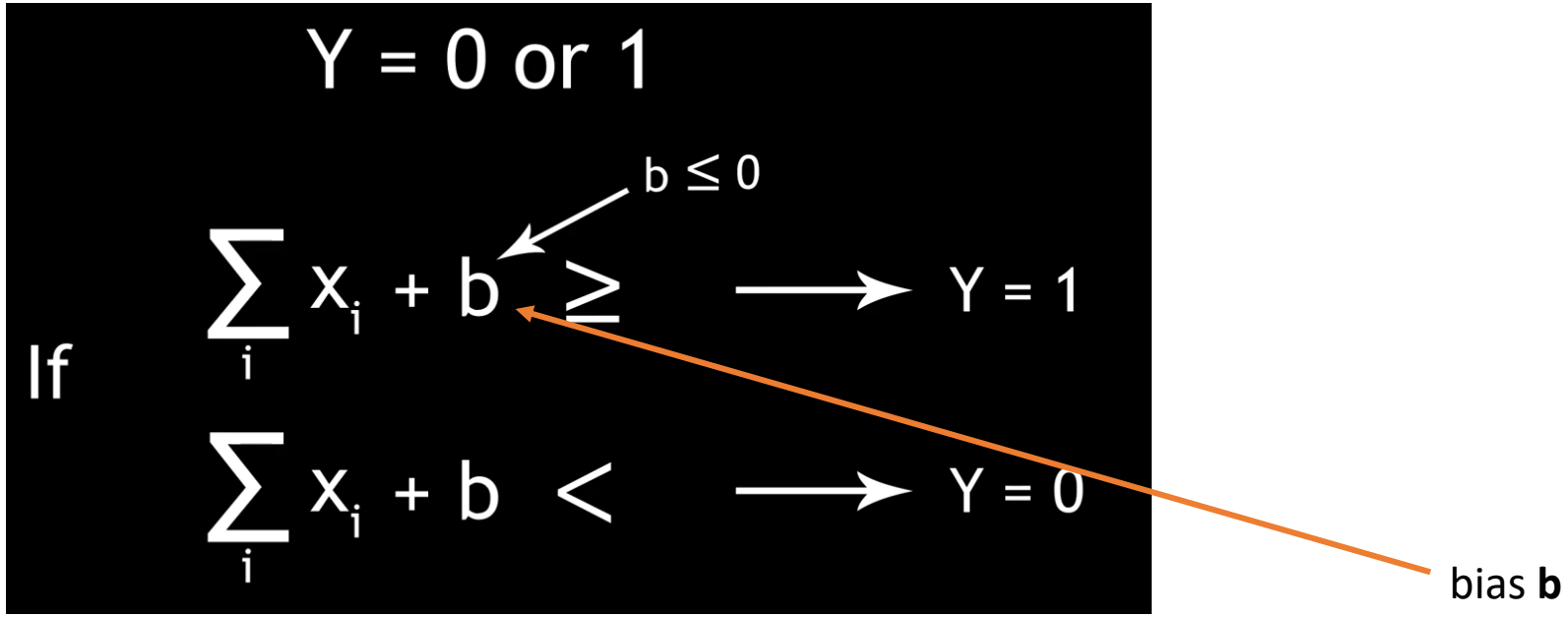
The threshold determines whether or not an input value will trigger an output value



https://www.youtube.com/watch?v=L_9OluD0nqw

Threshholds, or Bias Values

The threshold is expressed as a *bias*, which is a negative number added to the sum of the input values



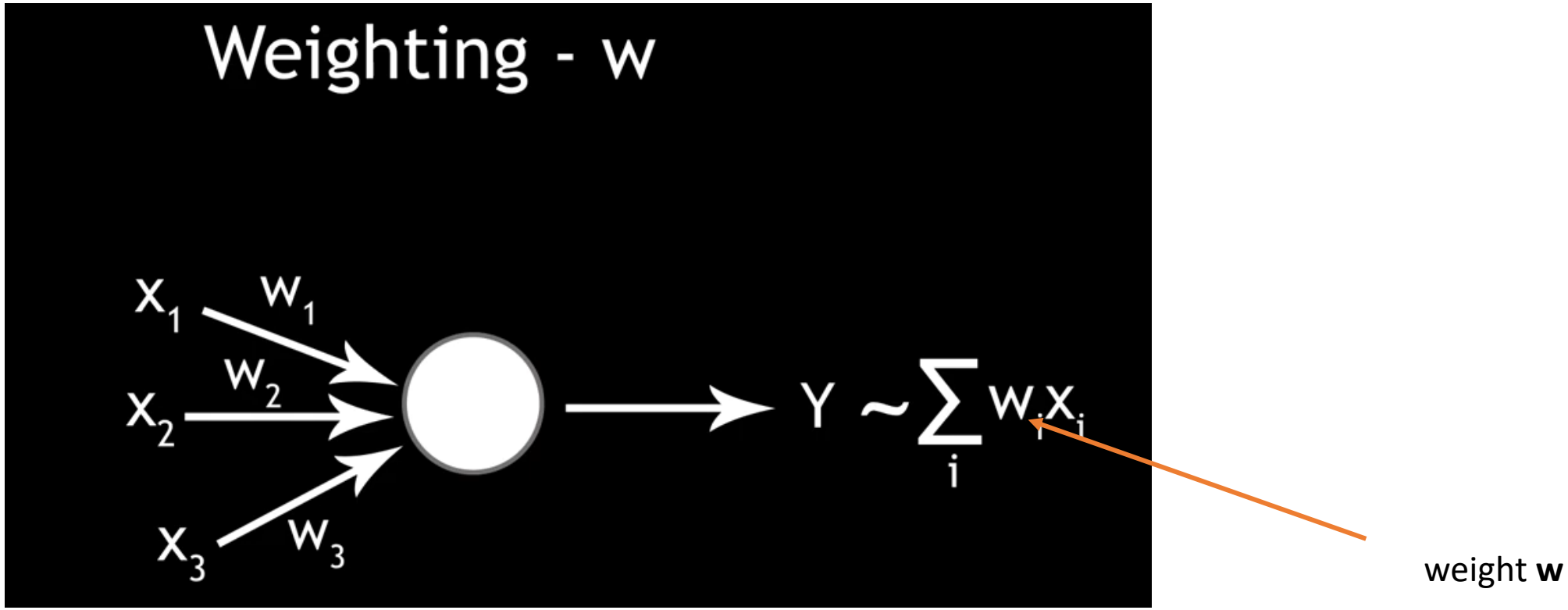
https://www.youtube.com/watch?v=L_9OluD0nqw

Activation Value

- The *activation value* of a neuron is a number generated from the input to the neuron. In the case of the perceptron, the activation is 0 or 1
- An *activation function* is the algorithm a neuron uses to generate its activation value from the input

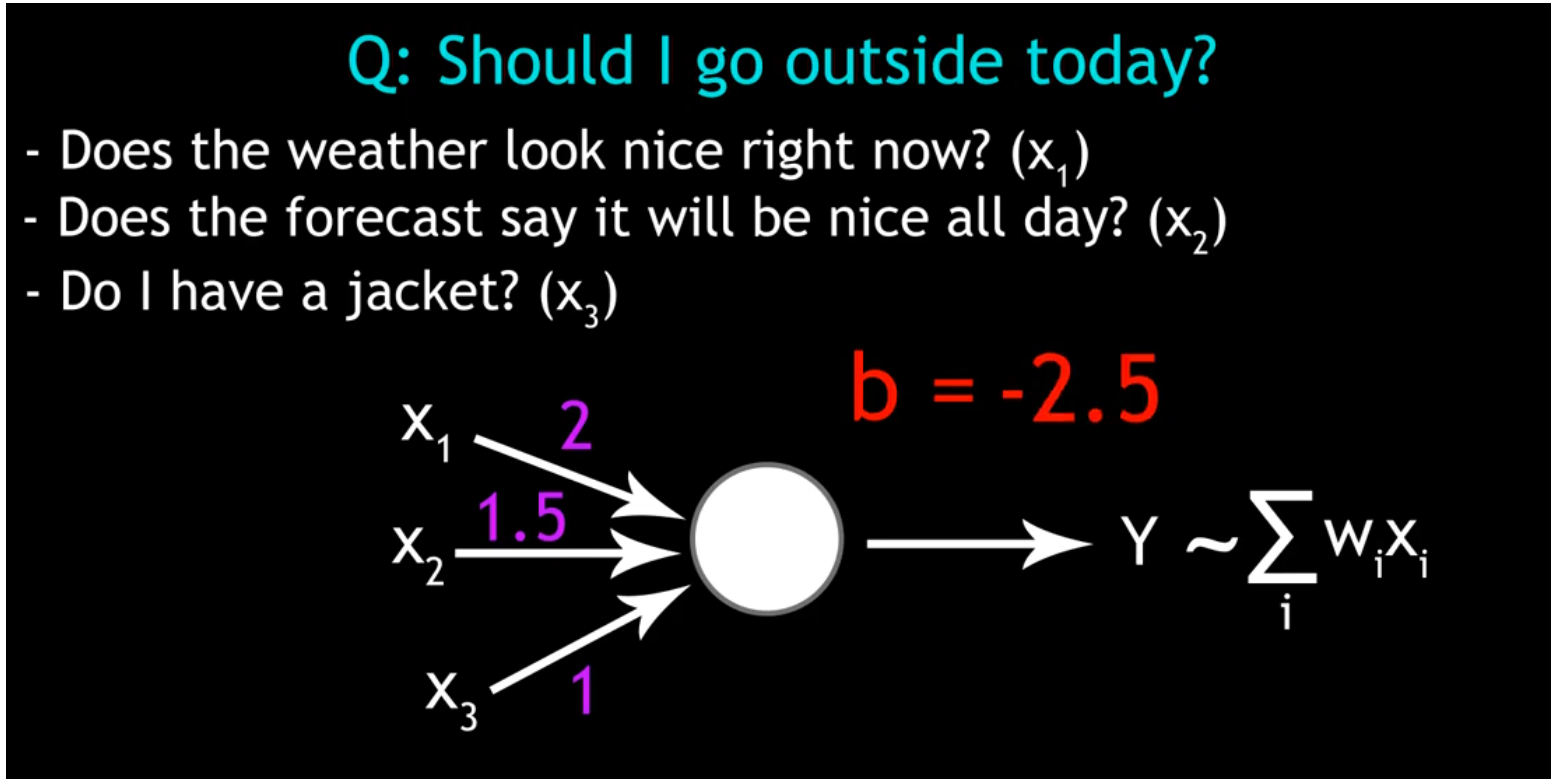
Weights

Weights alter how much influence each input value has on the neuron



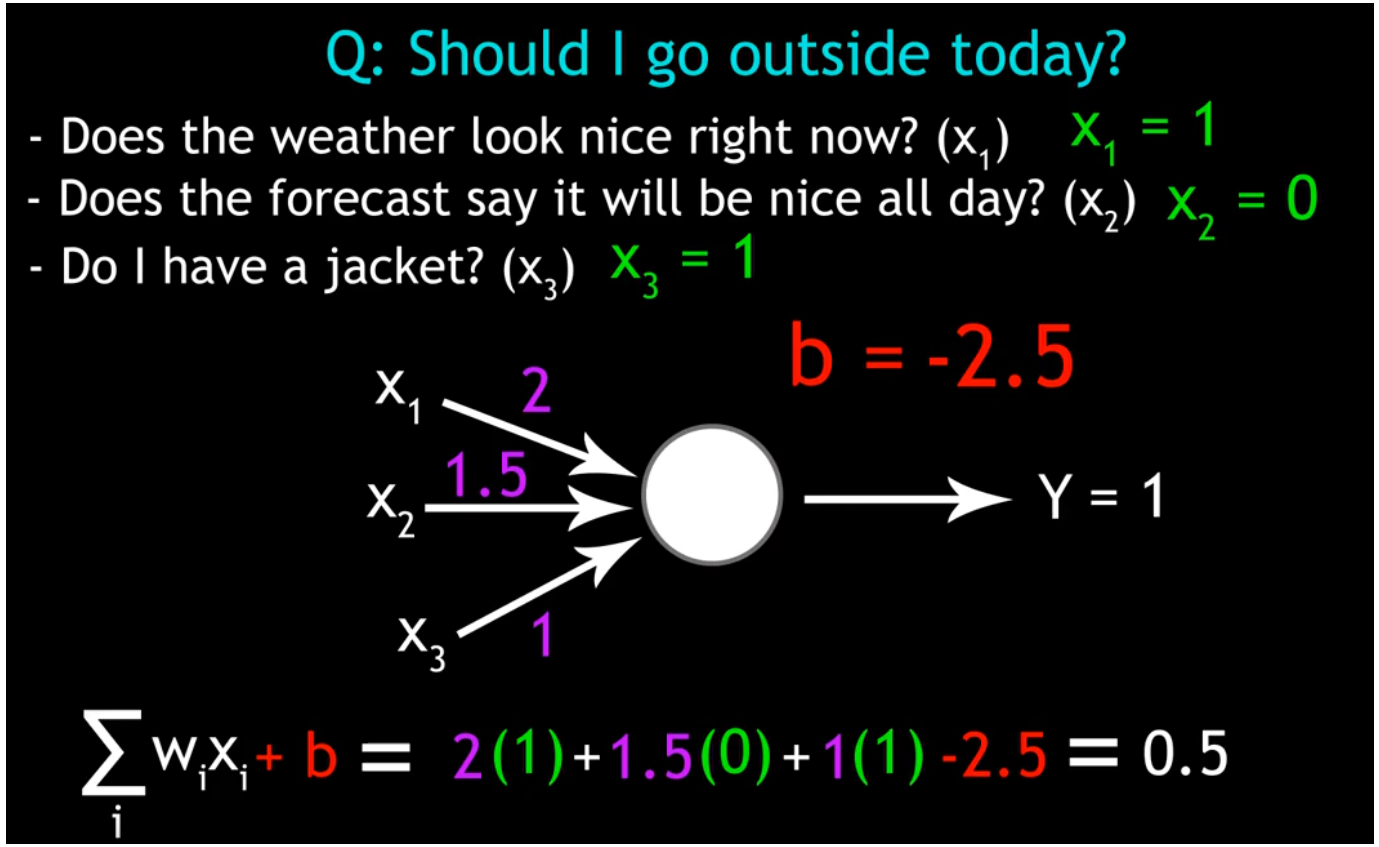
Calculating

See how the weight measures how important each factor is to me?



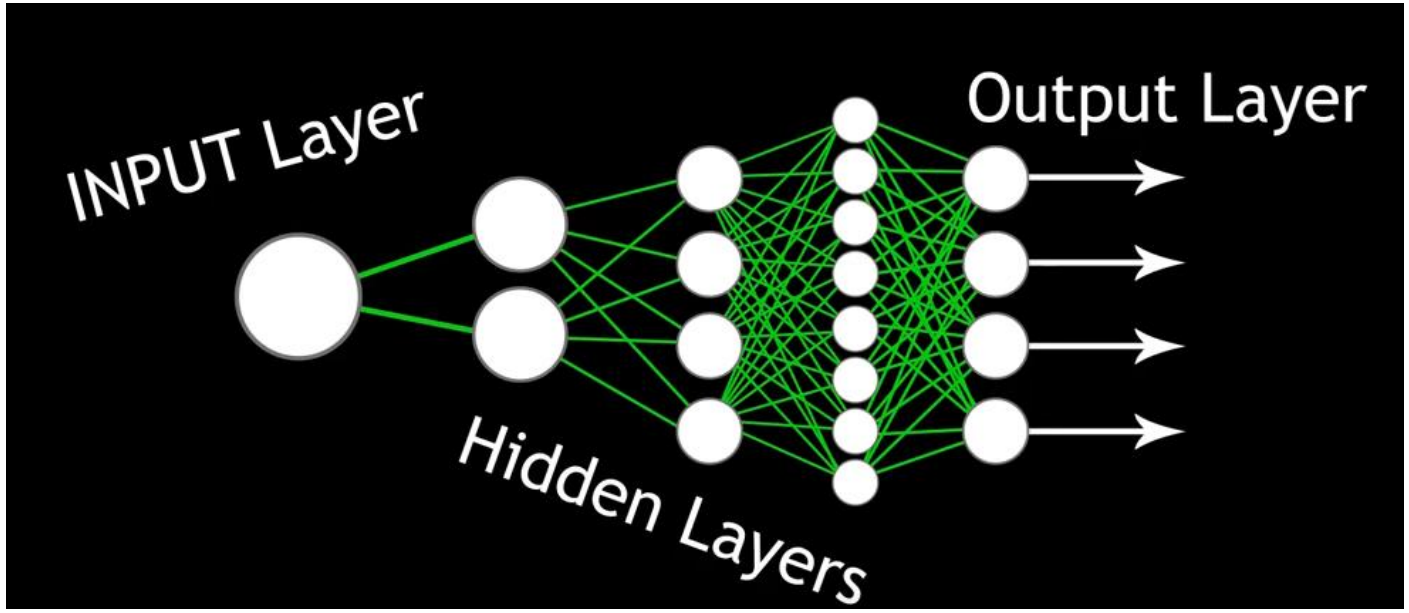
Calculating

- We collect the sum of the weight*value for each input and add the bias to produce the output.



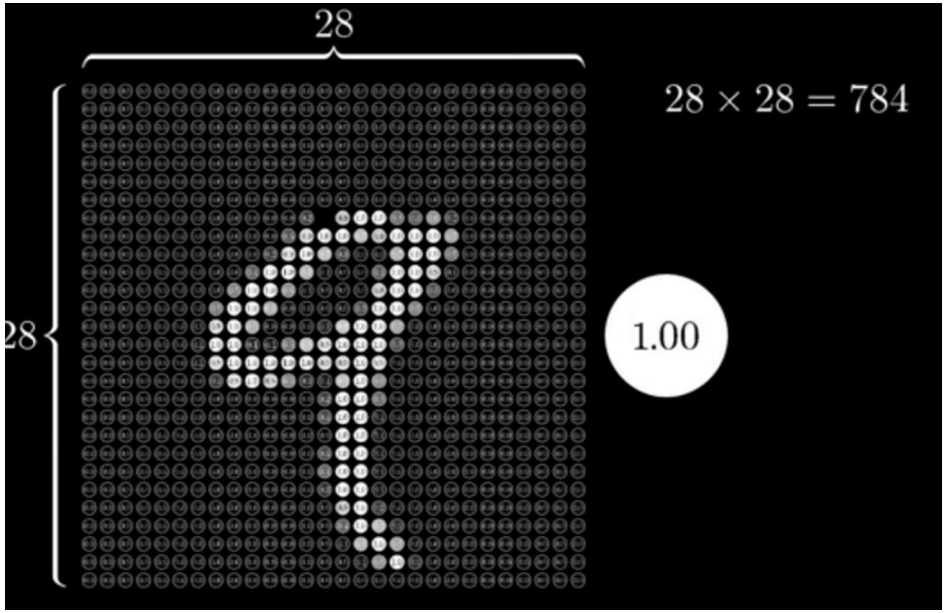
Networks of Perceptrons

These networks use the output from one set of neurons (called a *layer*) as the input for the next set of neurons.



Recognition Tasks

Example



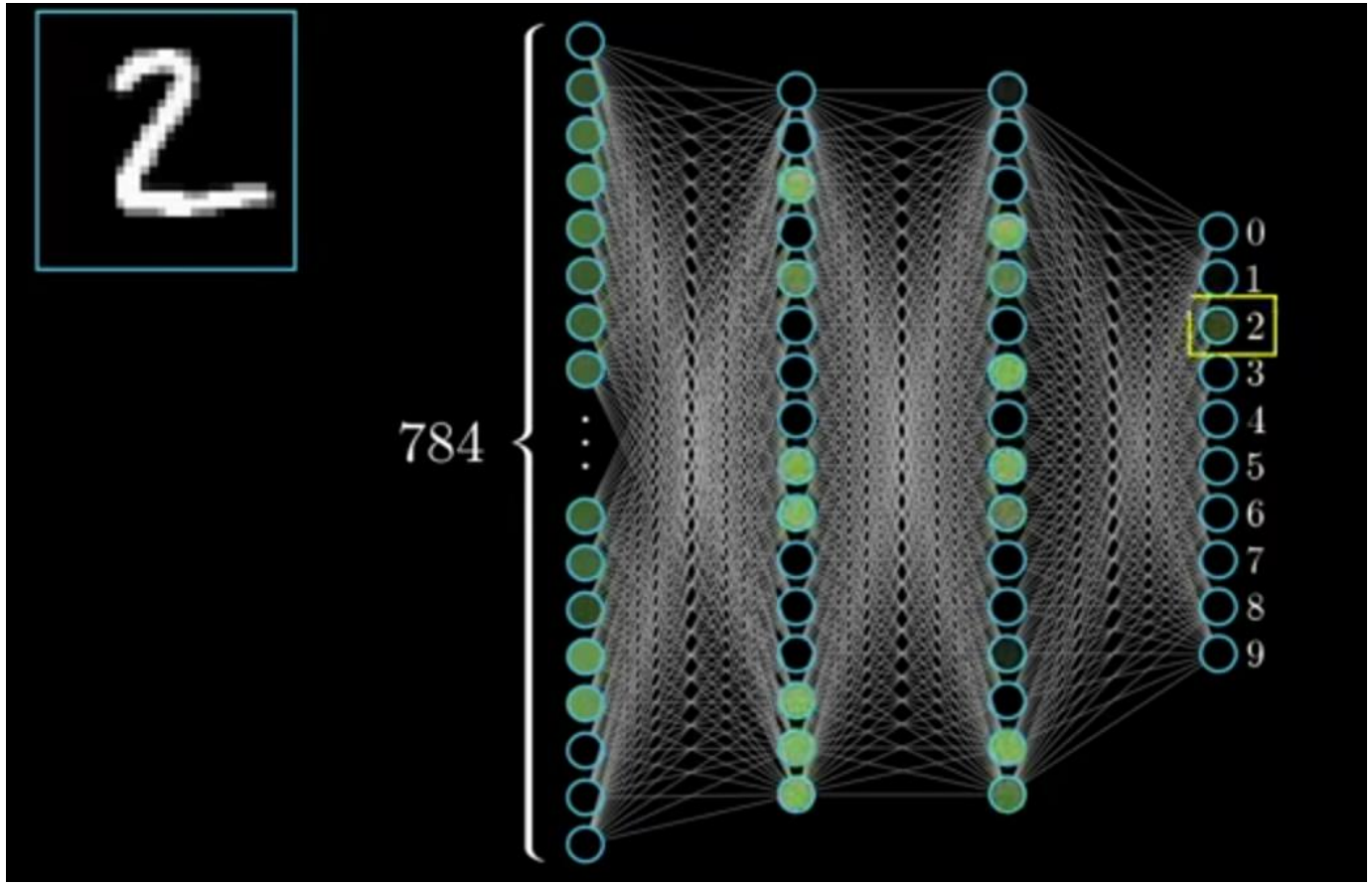
The input here is a layer of 781 neurons

https://www.youtube.com/watch?v=aircAruvnKk

Recognition Tasks

Activations are sent from layer to layer to produce a set of values in the output layer.

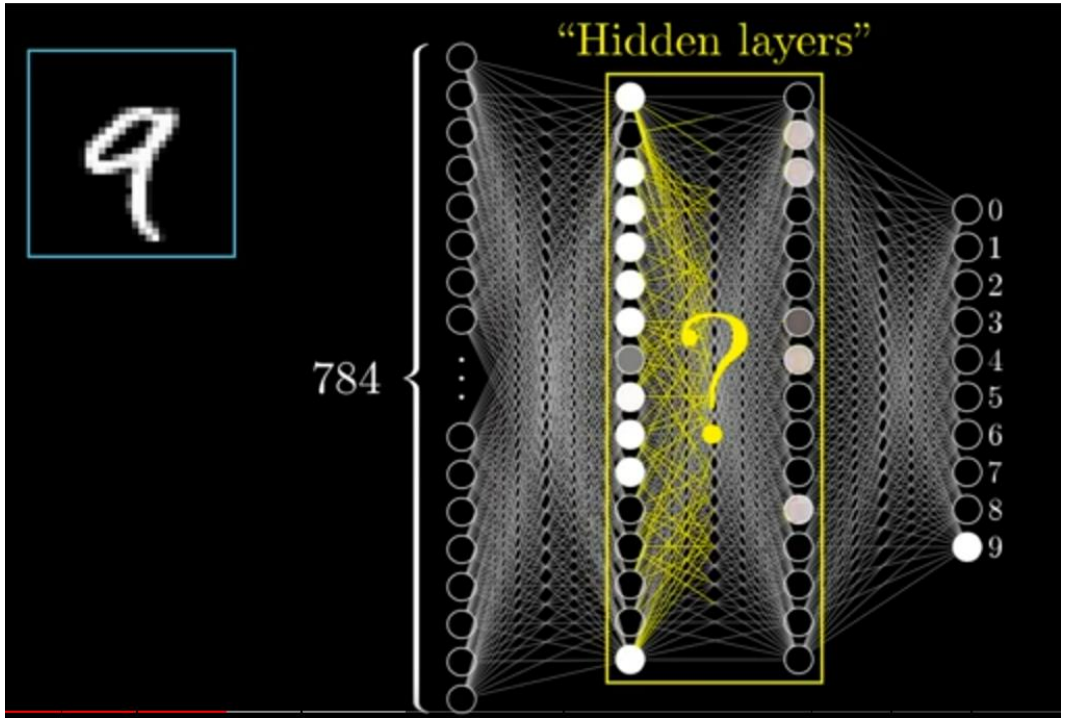
In this case we have an image as input, and a set of digits as an output.



https://www.youtube.com/watch?v=aircAruvnKk

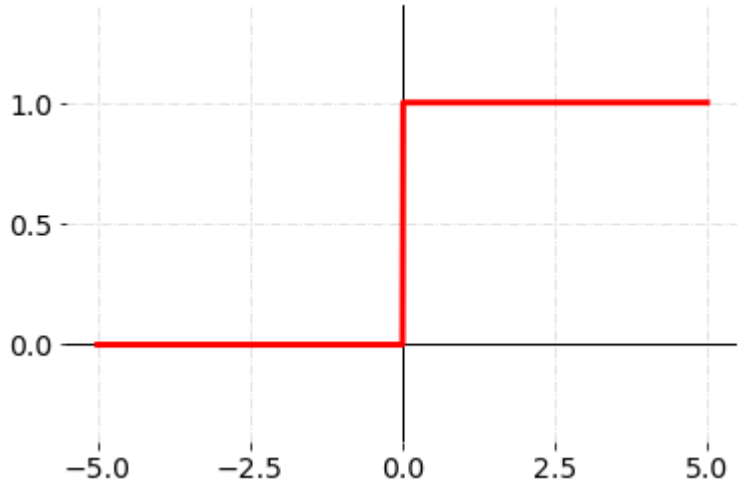
Layers

- In a network, activations (ie, the values of individual neurons) in one layer determine the activations in the next layer
- Notice the sizes of the layers are a bit arbitrary



Two Problems With Perceptrons

- They jump from 0 to 1, which is a big jump
- The weights have to be calculated manually



Your basic activation function has a value of 1 or 0. This is a *binary step* activation function.

https://towardsdatascience.com/getting-to-know-activation-functions-in-neural-networks-125405b67428

Sigmoid Neuron

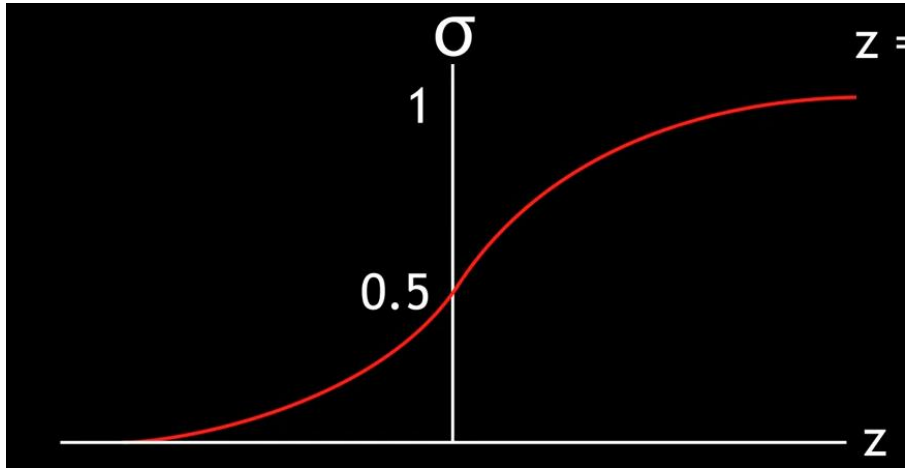
- Replaces perceptron neurons
- Output is calculated in the same way

$$z = \sum_i w_i x_i + b$$

- Then used in the sigmoid function

$$\text{Sigmoid} \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

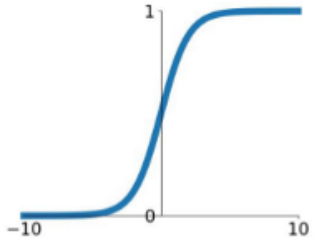
- The result, instead of being 1 or 0, is a smooth slope



Other Activation Functions

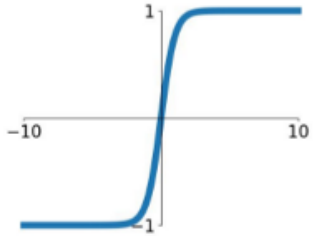
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



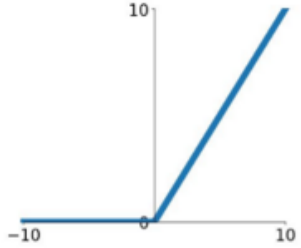
tanh

$$\tanh(x)$$



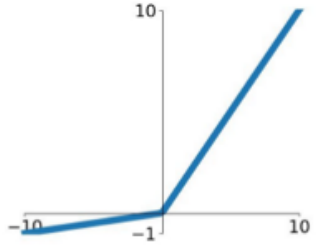
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

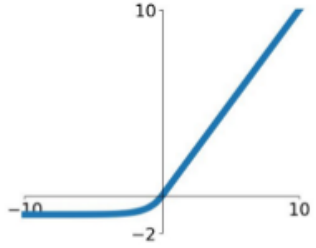


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

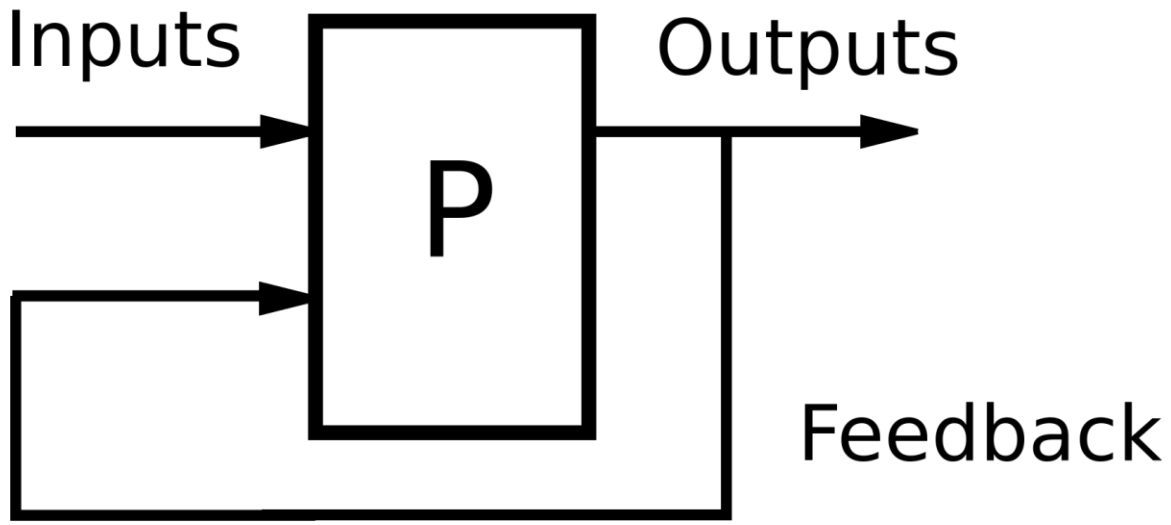
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



https://en.wikipedia.org/wiki/Activation_function
https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092
https://mlfromscratch.com/activation-functions-explained/

Backpropagation

- Adjusts the weights of a neural network
- Uses training examples to ‘train’ the network
- The result is then used to make predictions



https://en.wikipedia.org/wiki/Feedback

Cost Function

Is a measure of the difference between the desired outcome and the prediction

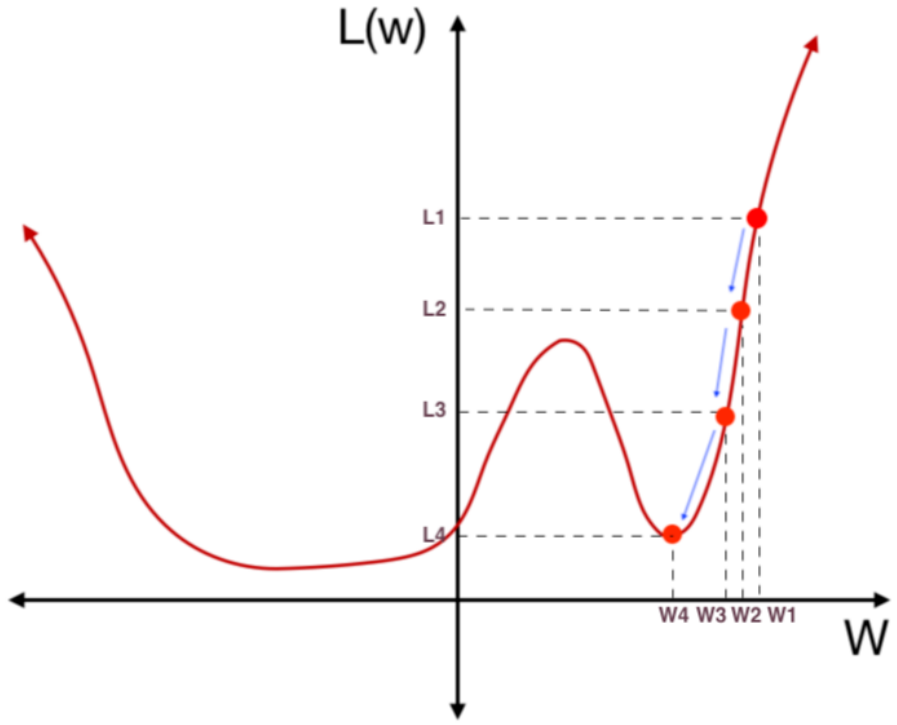
$$C = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Goal: Minimise C

Optimizing weights

- Reduce the value of C (the cost function) by adjusting weights
  - Calculate the *gradient* of C ∇C

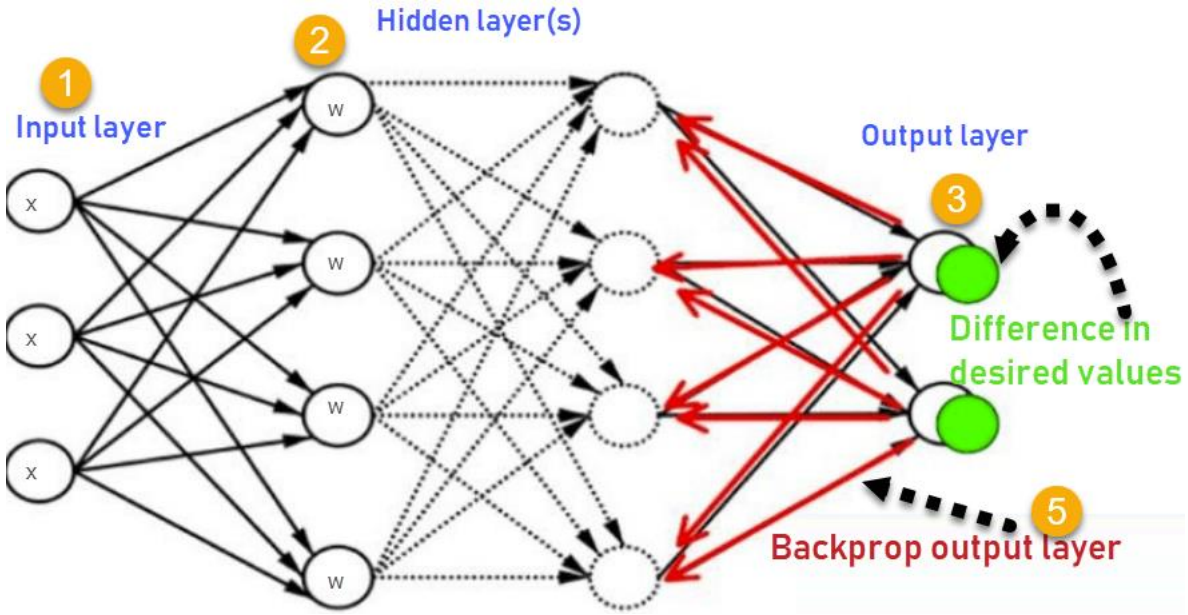
https://towardsdatascience.com/step-by-step-the-math-behind-neural-networks-490dc1f3cfd9



Propagation

- Guess the weights
- Provide input
- Identify the *error*

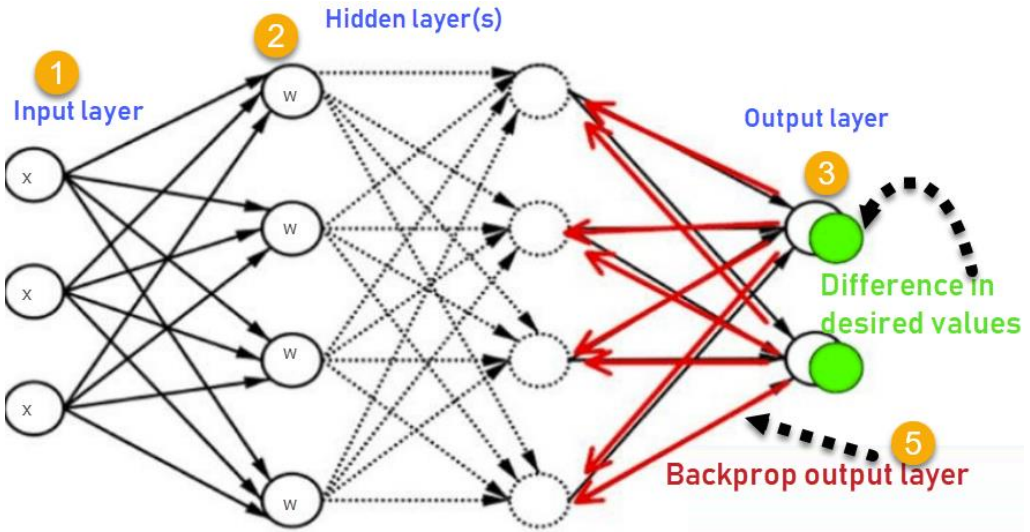
$$\varepsilon = y - \hat{y}$$



Back-Propagation

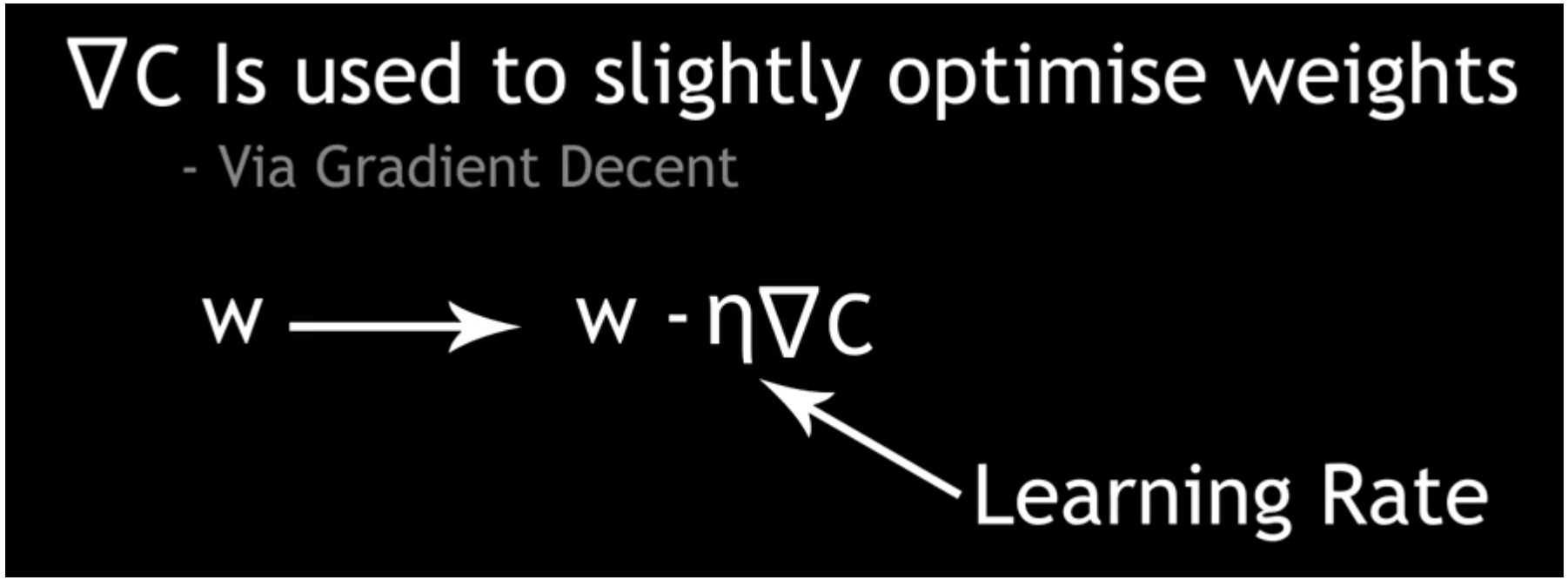
- The error for neurons in any particular error can be calculated as a function of the errors the next layer up

$$\text{Error} \quad \varepsilon_l = [w_{l+1}\varepsilon_{l+1}] \bullet \sigma(z)$$



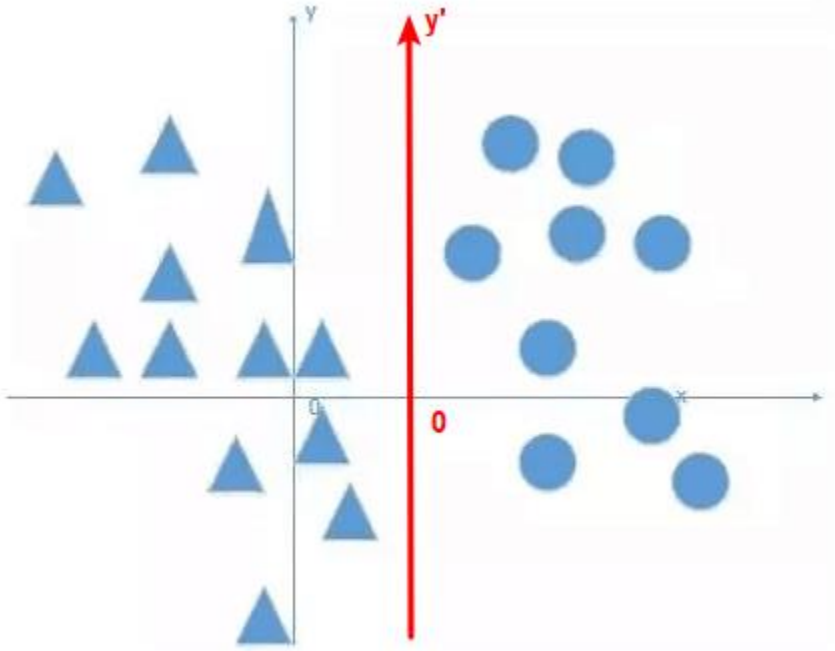
Adjusting Weights

- Calculate the *gradient* of C ∇C, as before



Adjusting Biases

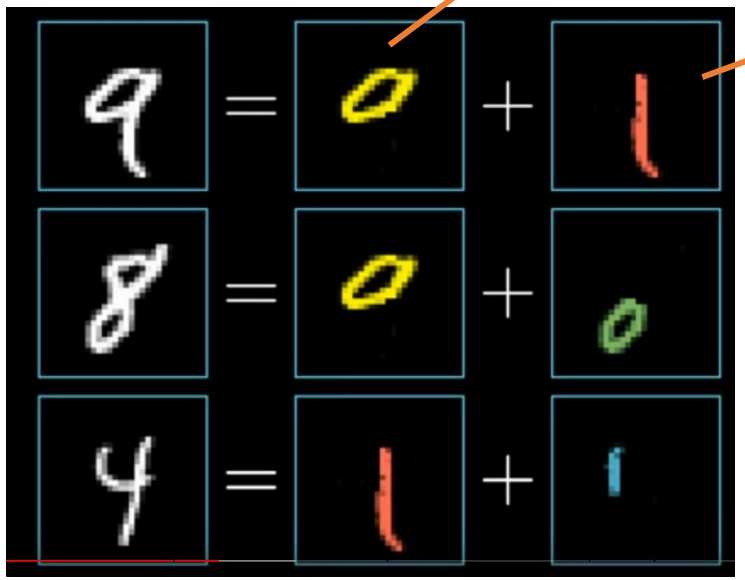
- You can also adjust the bias
- This establishes how active a neuron needs to be before it has an activation value

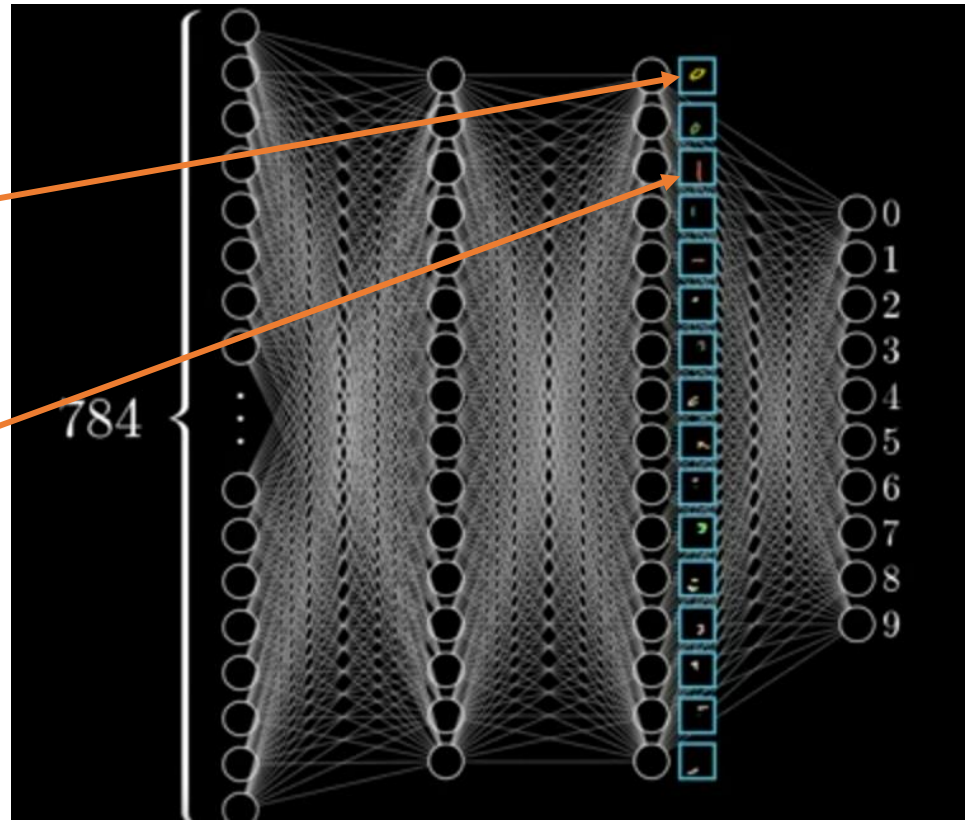


https://stackoverflow.com/questions/3775032/how-to-update-the-bias-in-neural-network-backpropagation
https://www.geeksforgeeks.org/effect-of-bias-in-neural-network/
https://www.tutorialexample.com/understand-bias-in-neural-network-why-using-bias-in-neural-network/

Feature Detection

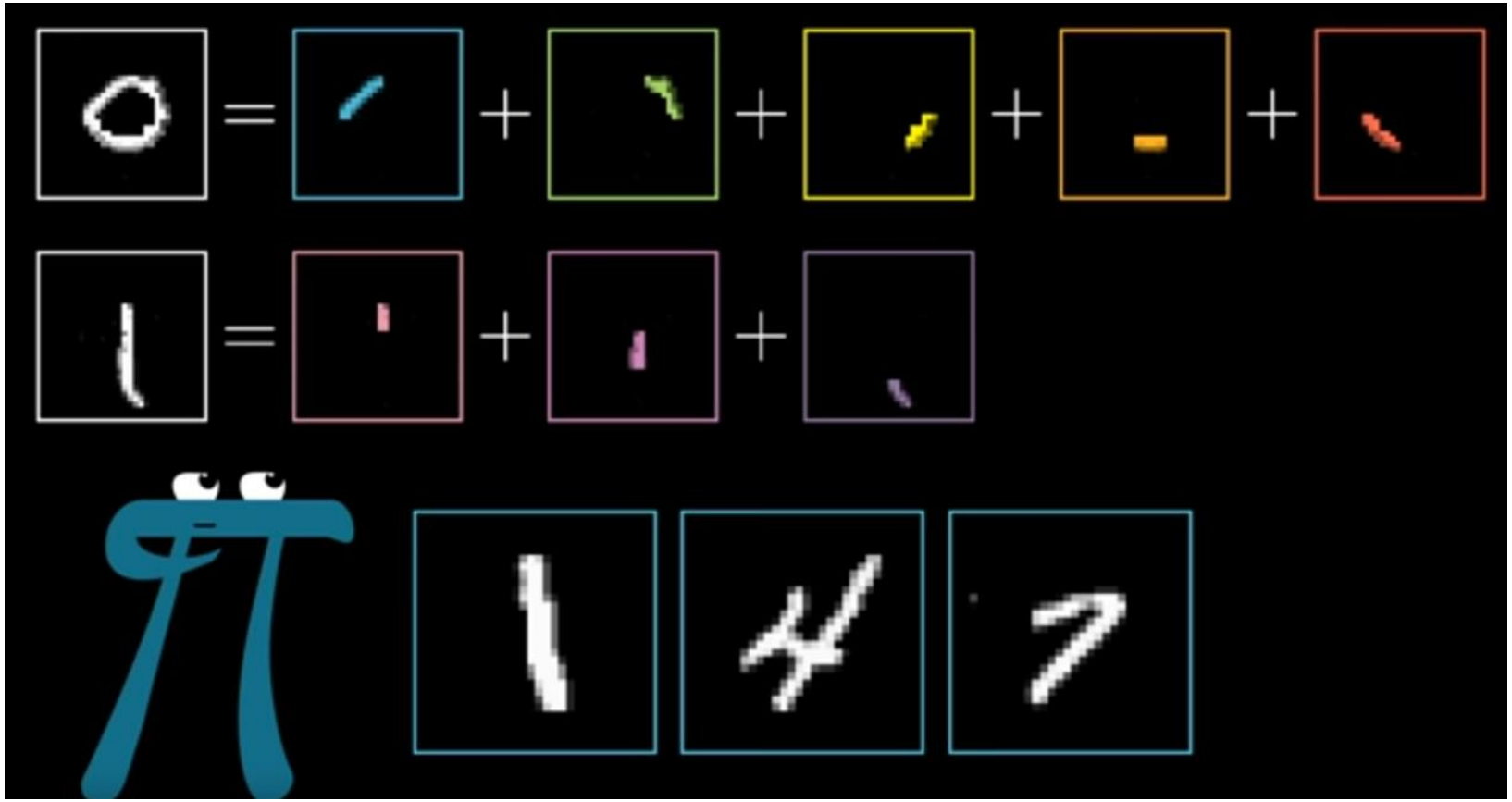
What do the layers do?





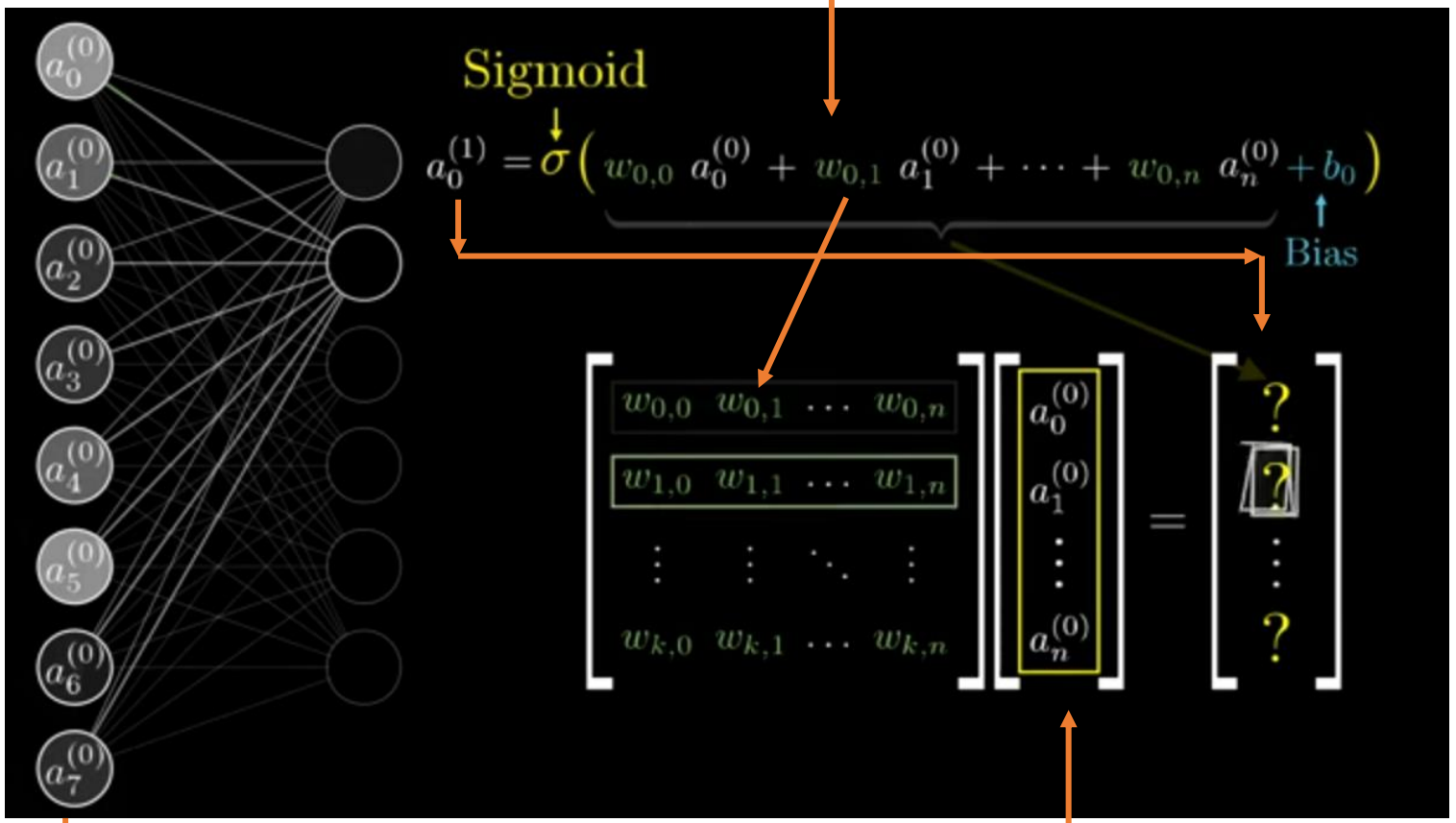
Edge Detection

Features are made of smaller parts

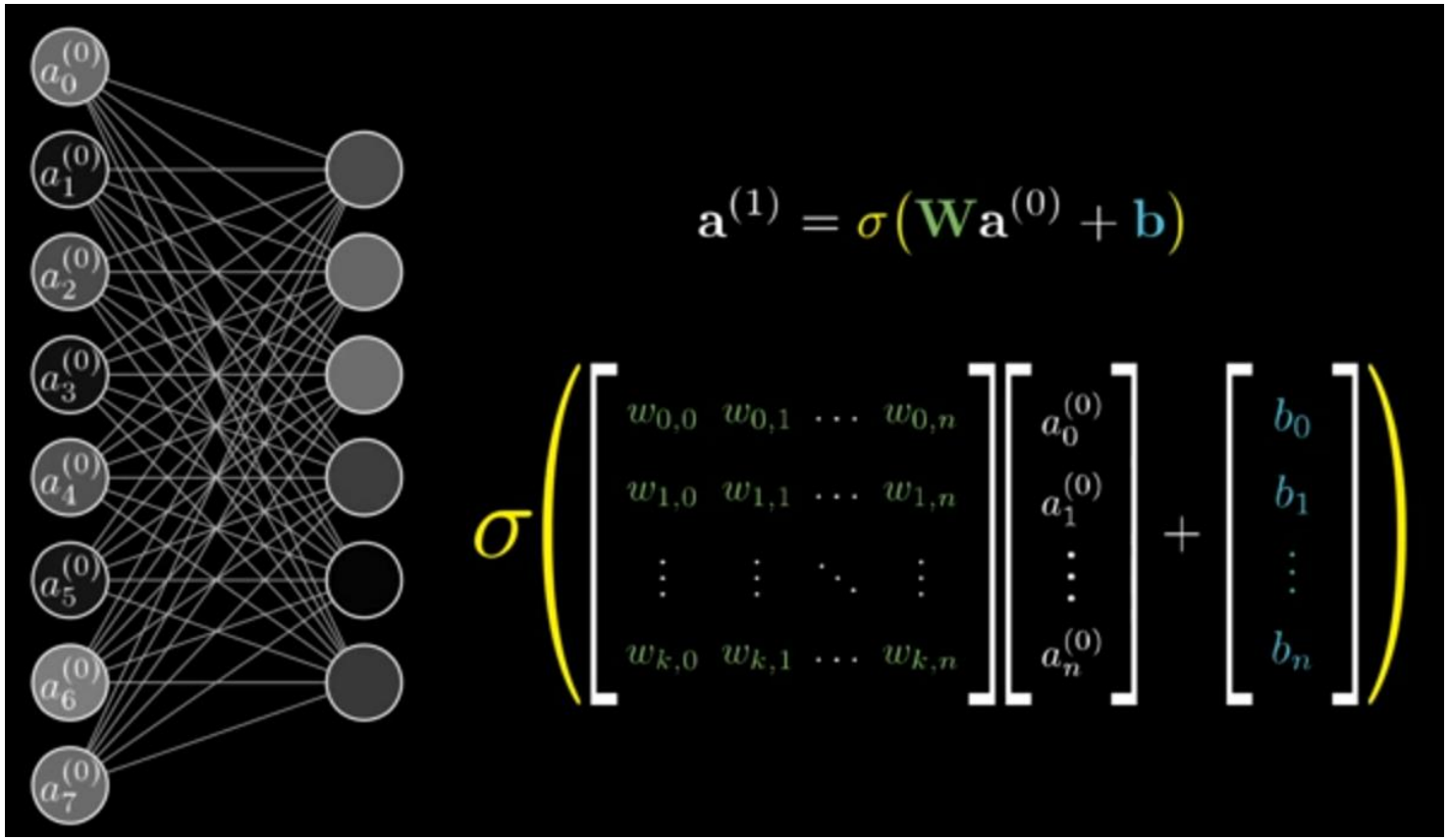


Calculations

- Calculations of the weights and values can be described using matrix algebra



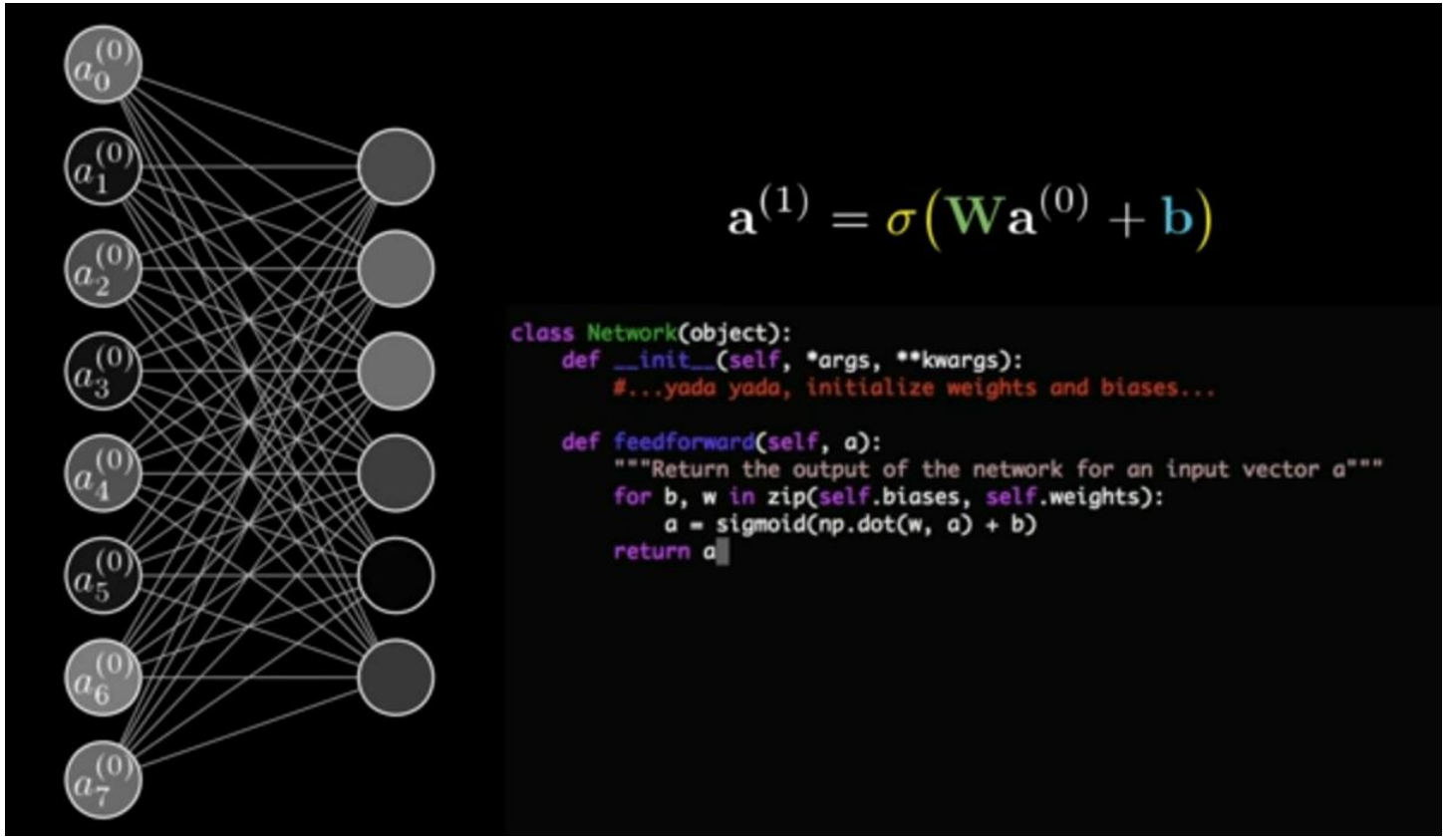
Expressions



Adding the bias…

Code

Code libraries are optimized for matrix multiplication



https://www.youtube.com/watch?v=aircAruvnKk

Iterations

- Each time we adjust all the weights, we call that an *epoch.*
- This is too large to do all at once, so it’s broken down into batches.
- The number of iterations is the number of batches needed to complete an epoch.