ADACS

ASTRONOMY DATA AND COMPUTING SERVICES

Astroinformatics school - "Rise of the machines"

Curtin University
THE CURTIN INSTITUTE FOR COMPUTATION

4 to 6 February 2019

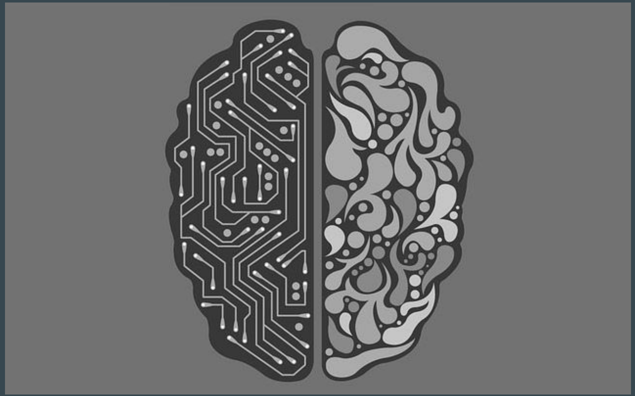
Presented by Rebecca Lange and Dan Marrable

Artificial Neural Networks

Curtin Institute for Computation

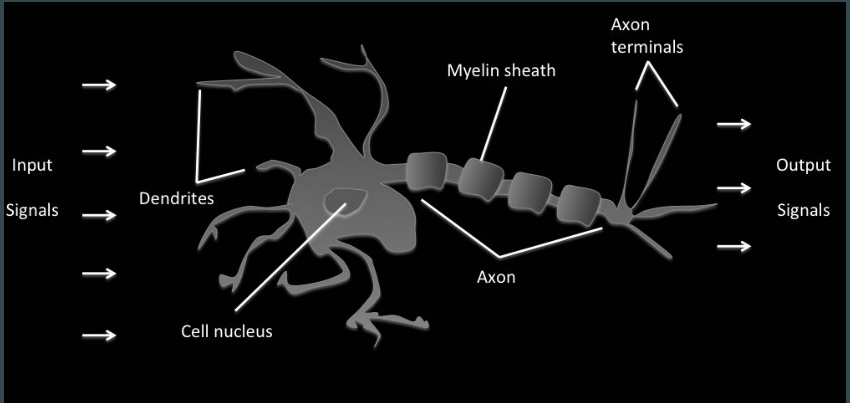
What are Artificial Neural Networks?

- Universal function approximators
- A series of linear equations (and activation functions) that the approximate nonlinear equations.
- Modelled loosely around how the brain's synapsis works.
- Generally used for classification tasks

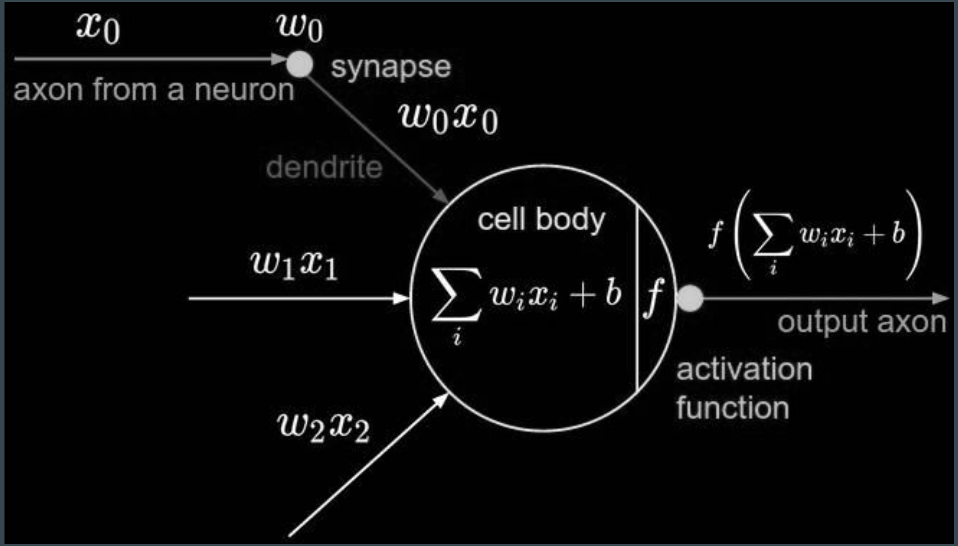


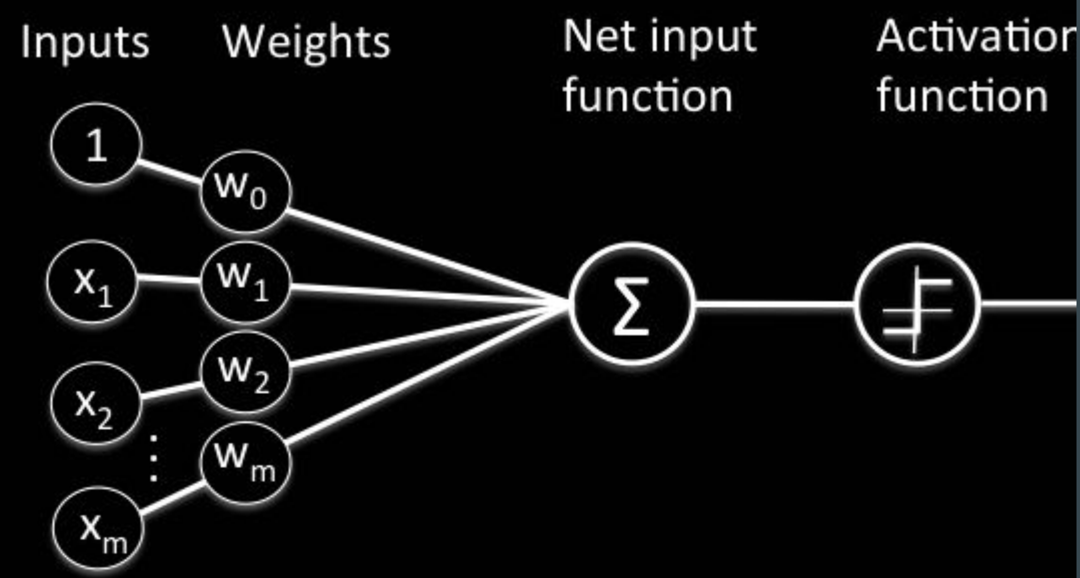
Neuron

Biological Inspiration



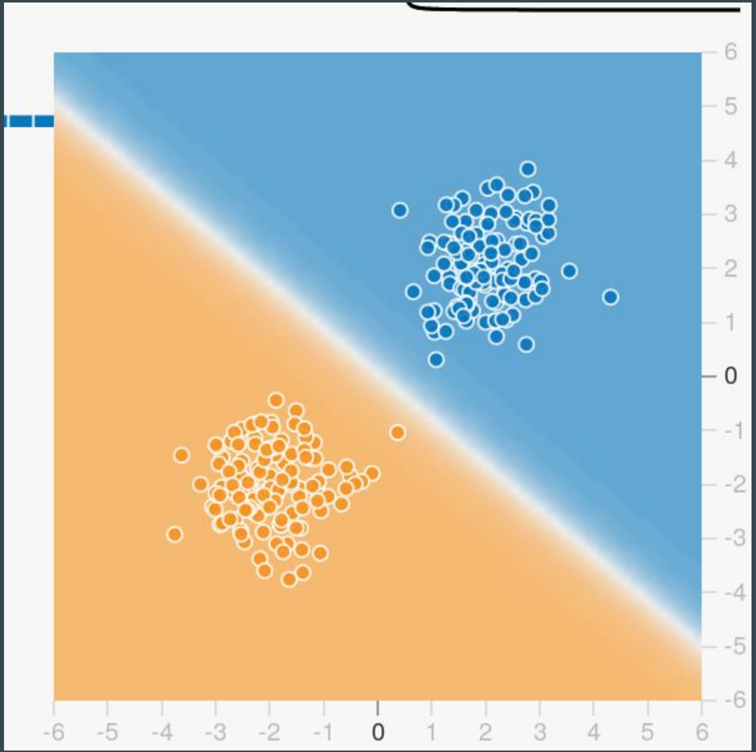
Schematic of a biological neuron.

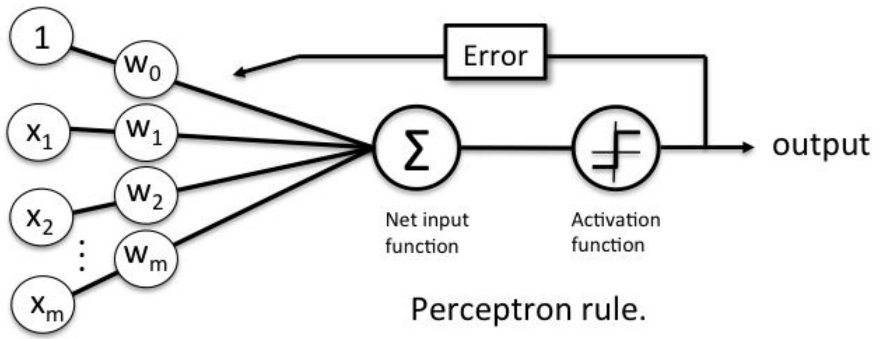




Schematic of Rosenblatt’s perceptron.

Single Perceptron



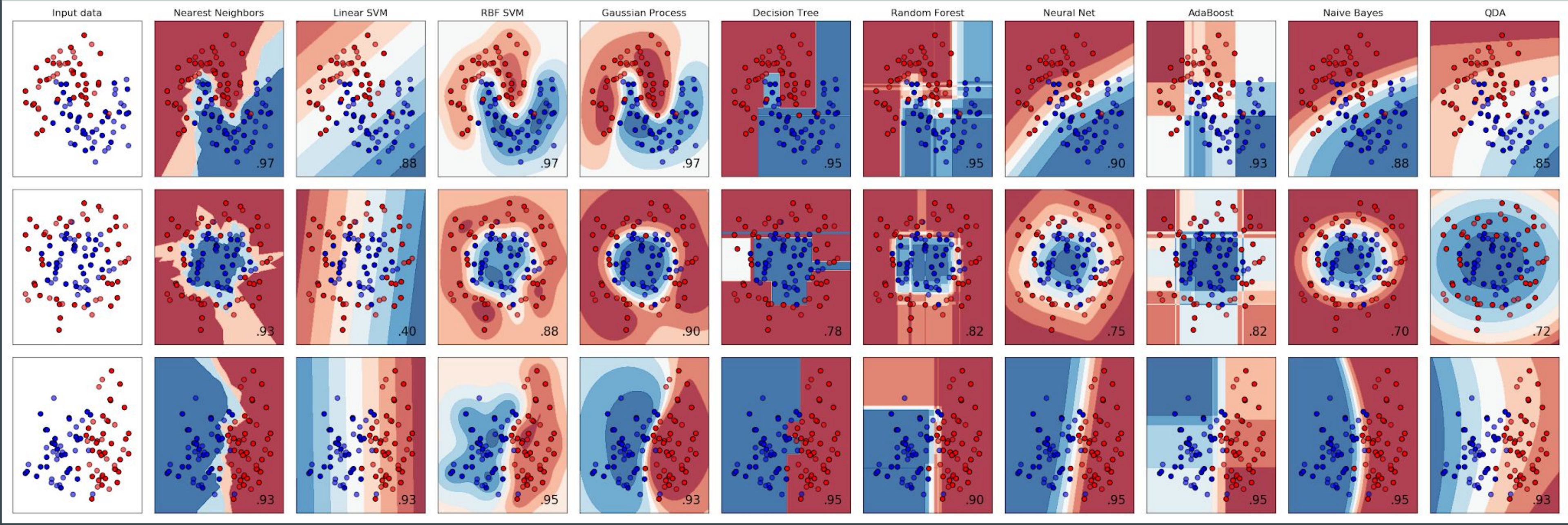


Perceptron rule.

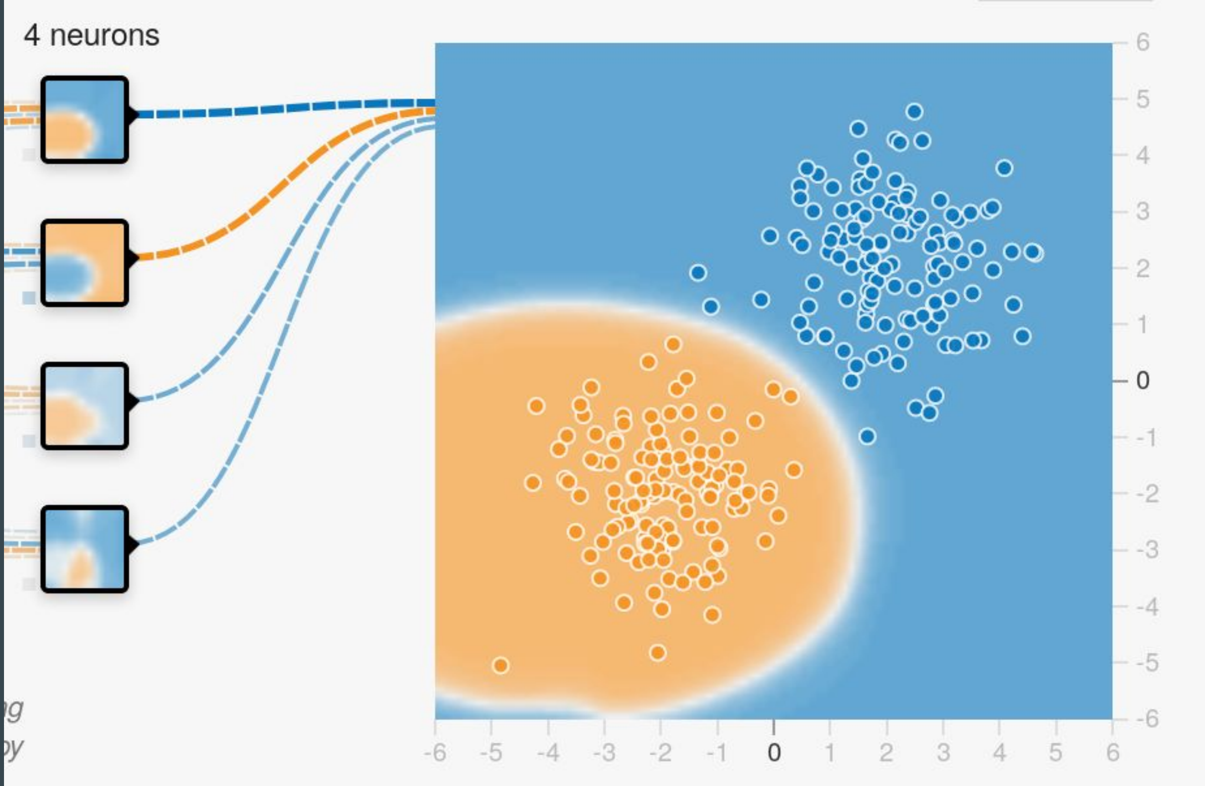
Limitations

A perceptron can only be used to linearly separable classes

<u>None of these cases are possible</u>



A Network of Neurons

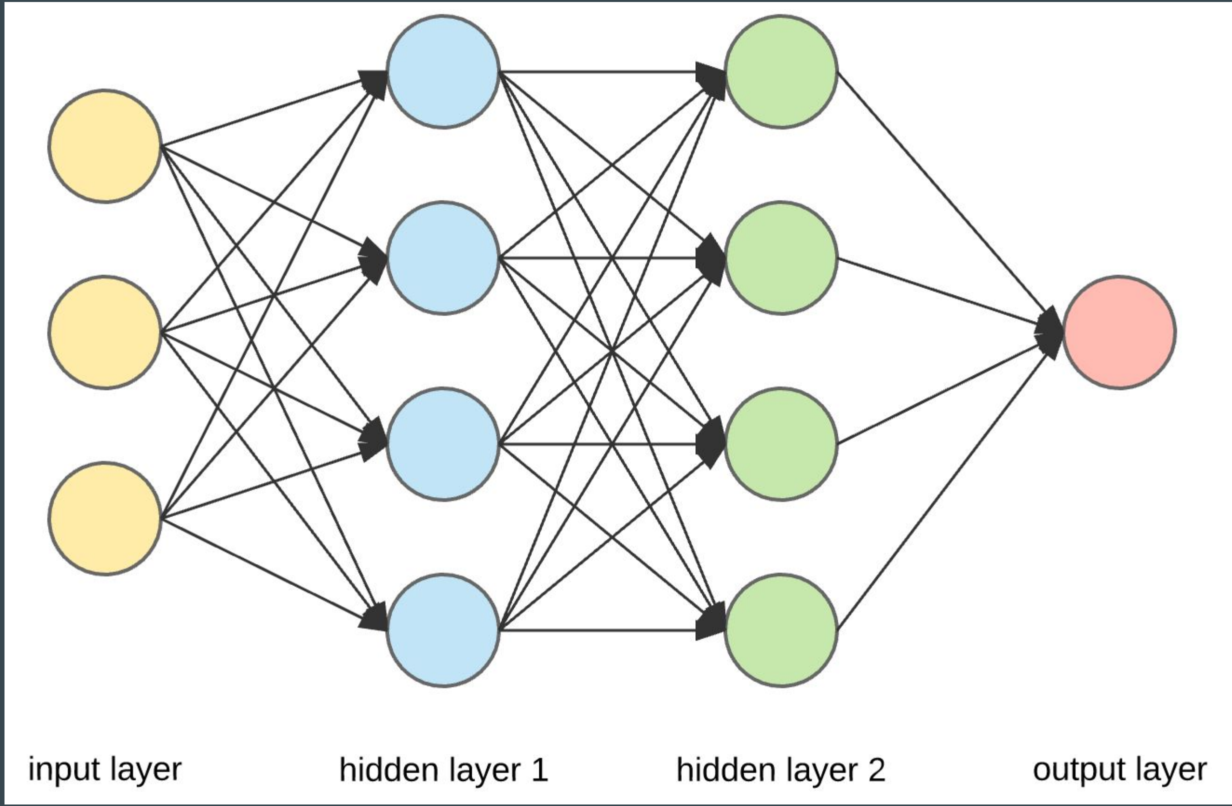


A Typical Neural Network

Input Layers takes in the data

Hidden layers have most of the training parameters

The output layer gives the classification



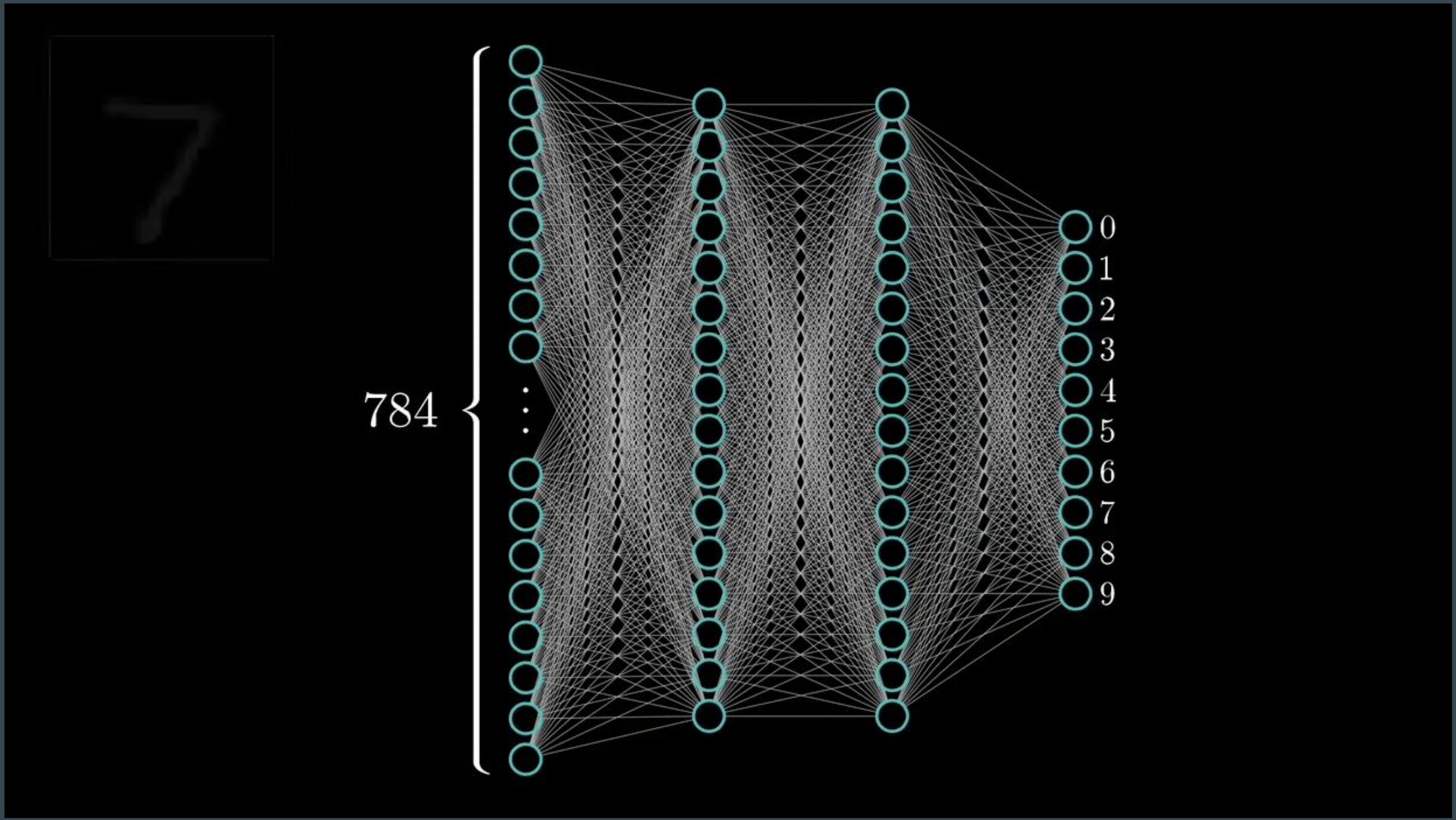
Example

Some combination of parameters will make our output layer activate correctly

We don't know what they are yet

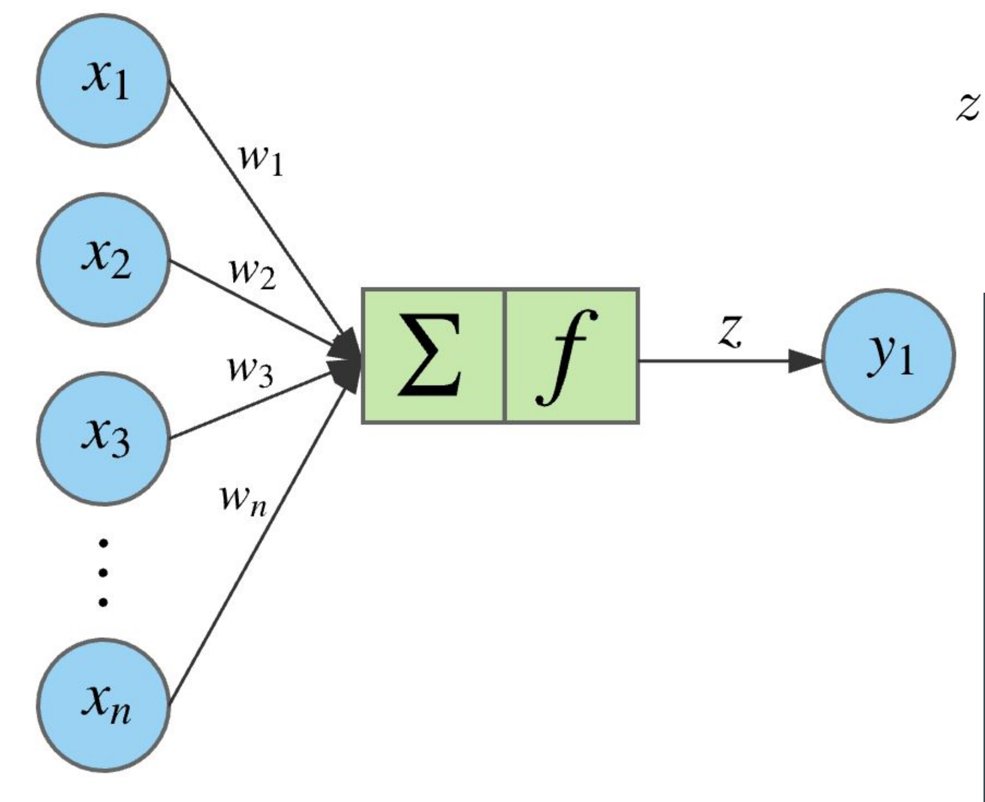
There are too many to randomly guess

We need an algorithm to find them



3Blue1Brown : https://youtu.be/aircAruvnKk

Mathematical Representation



$$z = f(b + x \cdot w) = f\left(b + \sum_{i=1}^{n} x_i w_i\right)$$

$$x \in d_{1 \times n},\ w \in d_{n \times 1},\ b \in d_{1 \times 1},\ z \in d_{1 \times 1}$$

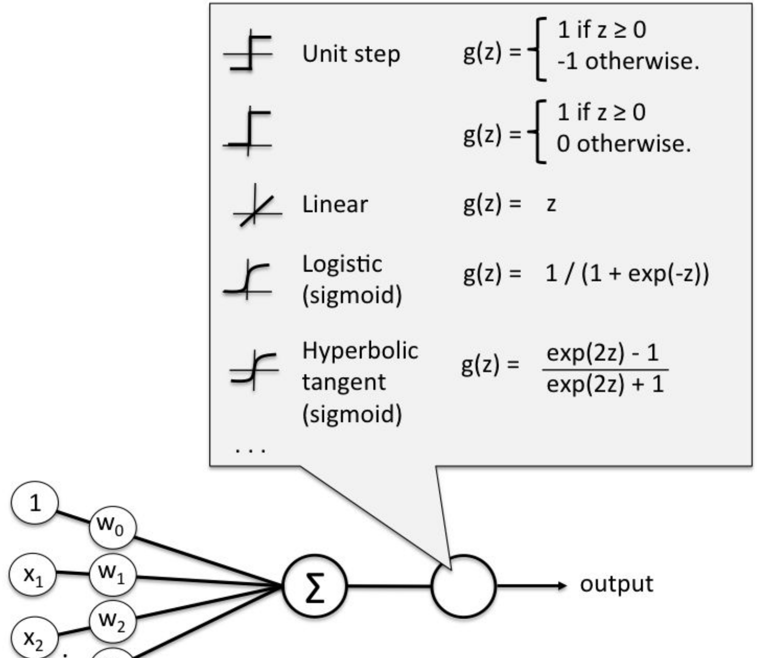
Multiply the inputs by a **weight**

Sum the result

Add a **bias**

Outputs are passed to an **activation function**

Activation Function

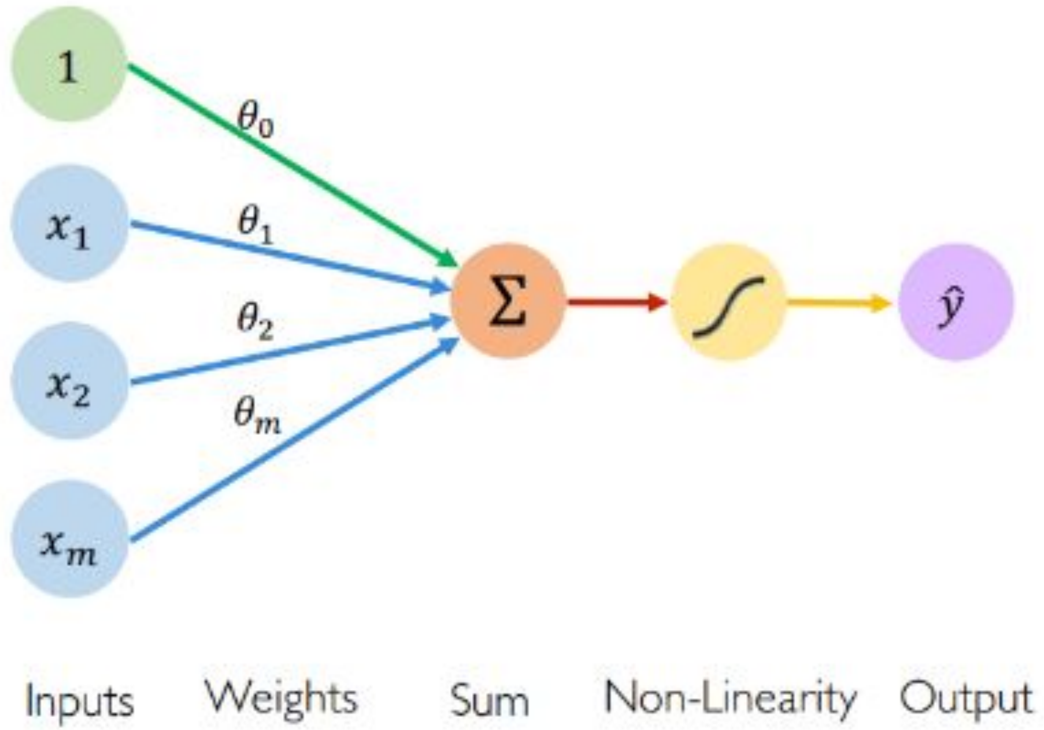


A selection of commonly used activation functions for artificial neurons.

Activation functions give the neurons a non-linear response

Sigmoids converge to either 1 or 0, but are slow to train and ‘kill’ gradients

If unsure, the rule of thumb is, use ReLU for hidden layers and sigmoid for the output layer



Output

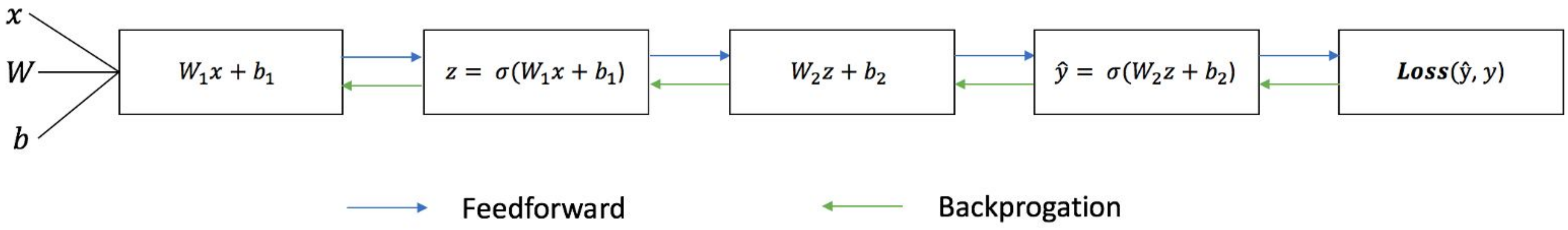
Linear combination of inputs

$$\hat{y} = g\left(\theta_0 + \sum_{i=1}^{m} x_i\,\theta_i\right)$$

Non-linear activation function

Bias

Training our ANN



There are different loss functions.

Problem type	Last-layer activation	Loss function
Binary classification	sigmoid	binary_crossentropy
Multiclass, single-label classification	softmax	categorical_crossentropy
Multiclass, multilabel classification	sigmoid	binary_crossentropy
Regression to arbitrary values	None	mse
Regression to values between 0 and 1	sigmoid	mse or binary_crossentropy

Make a prediction by making a forward propagation

Calculate the Loss

Back propagation to update the training parameters

Training our ANN

Full vectorised derivation
(https://www.coursera.org/specializations/deep-learning)

x, W, b → $W_1x + b_1$ ⇄ $z = \sigma(W_1x + b_1)$ ⇄ $W_2z + b_2$ ⇄ $\hat{y} = \sigma(W_2z + b_2)$ ⇄ $\boldsymbol{Loss}(\hat{y}, y)$

→ Feedforward

← Backprogation

$$Loss(y, \hat{y}) = \sum_{i=1}^{n} (y - \hat{y})^2$$

$$\frac{\partial\, Loss(y,\hat{y})}{\partial W} = \frac{\partial Loss(y,\hat{y})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z} * \frac{\partial z}{\partial W} \quad \text{where } z = Wx + b$$

$$= 2(y - \hat{y}) * \text{derivative of sigmoid function} * x$$

$$= 2(y - \hat{y}) * \text{z(1-z)} * x$$

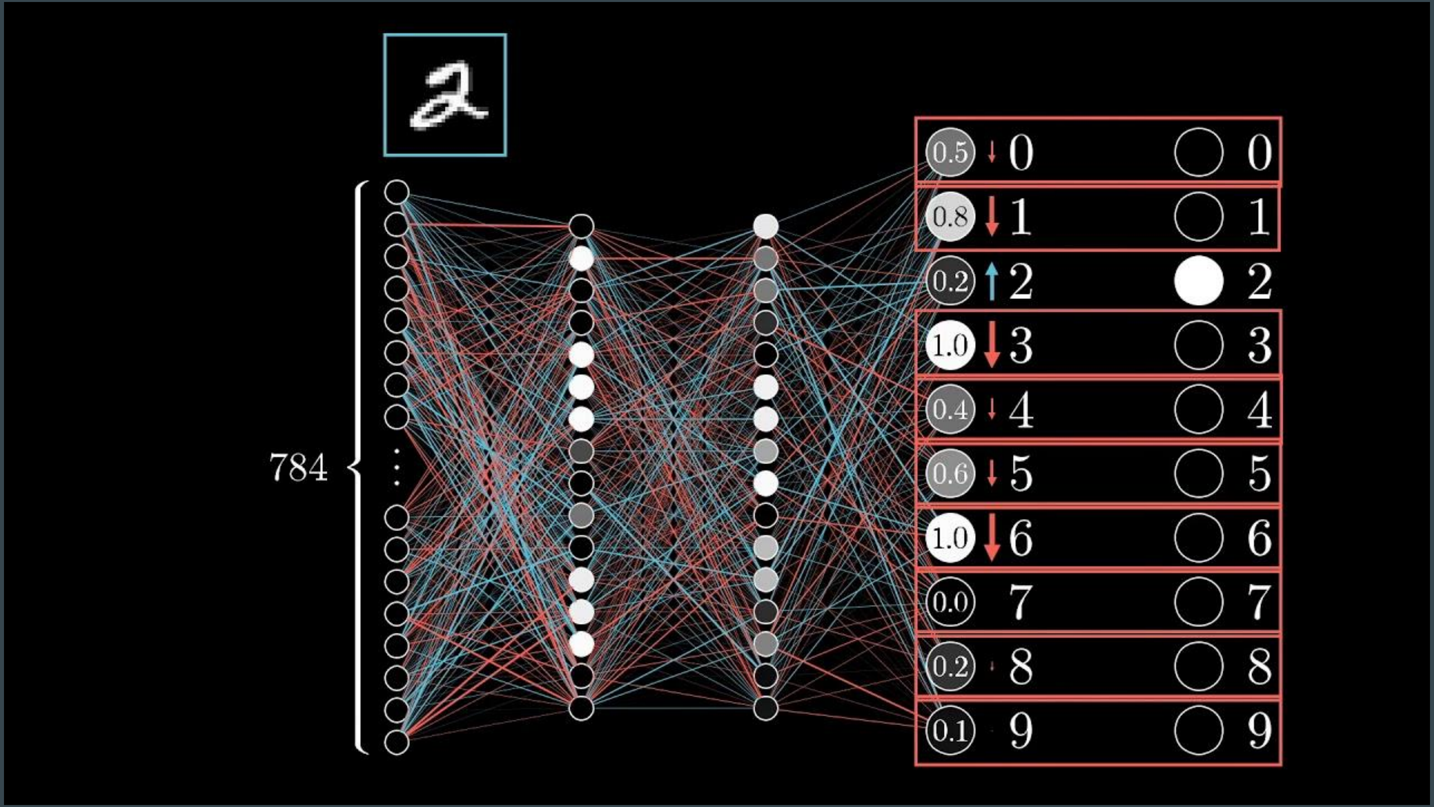
$$\Delta W_i = \eta \frac{\partial E}{\partial w_i}$$

η is the learning rate

$$W_i = W_i + \Delta W_i$$

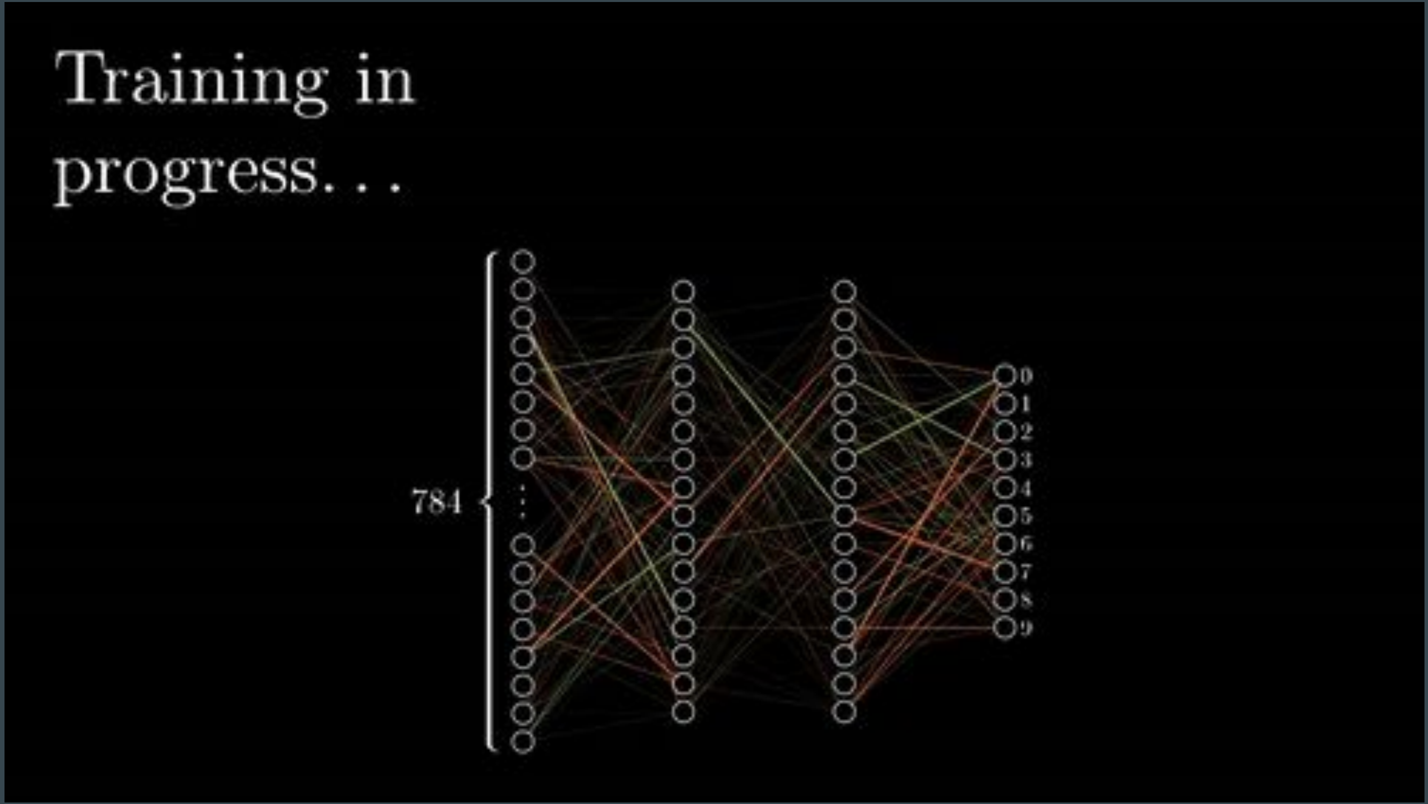
3Blue1Brown : https://youtu.be/aircAruvnKk

Backpropagation

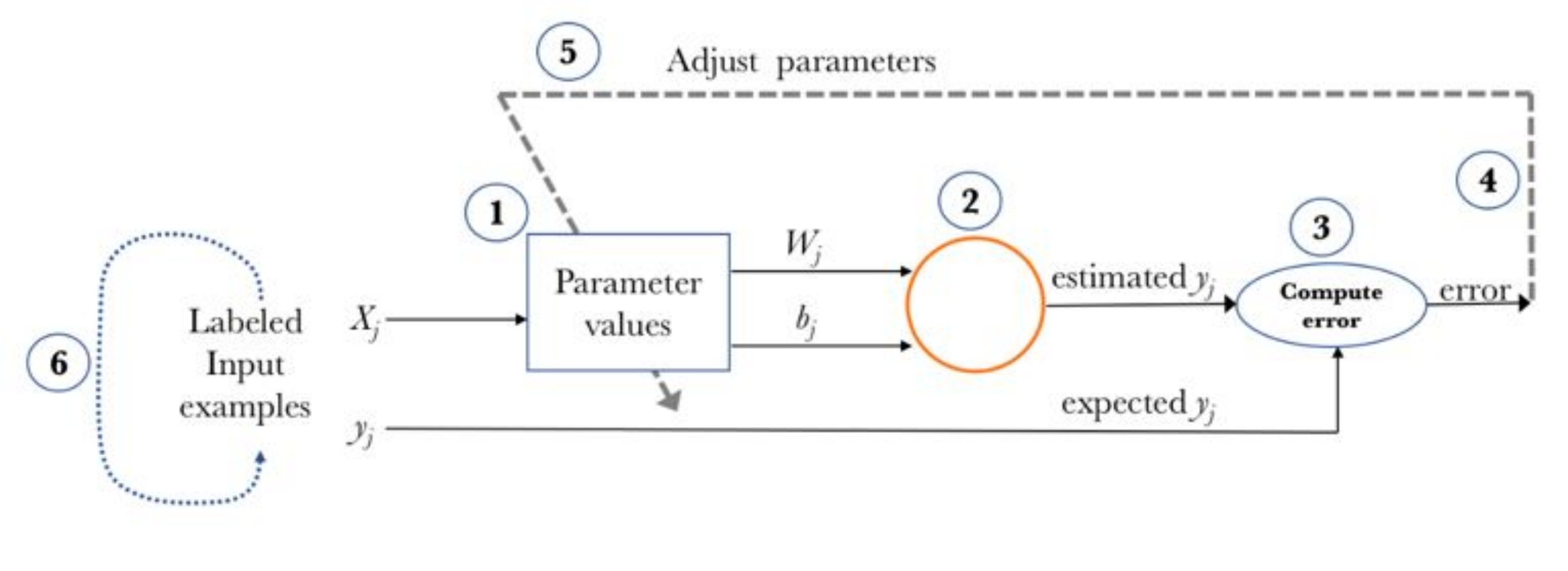


Training the model

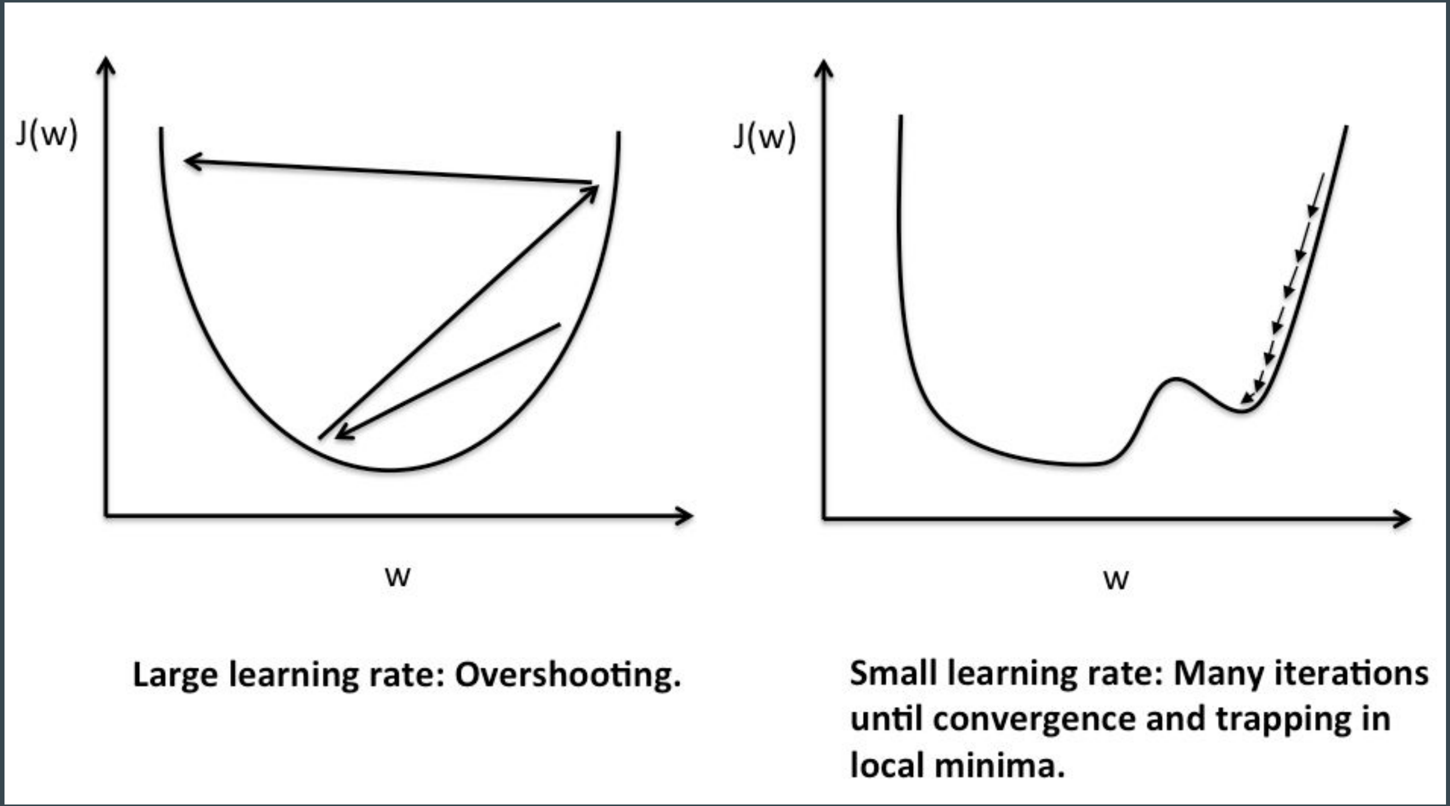
3Blue1Brown : https://youtu.be/aircAruvnKk



Optimiser - predictor - corrector

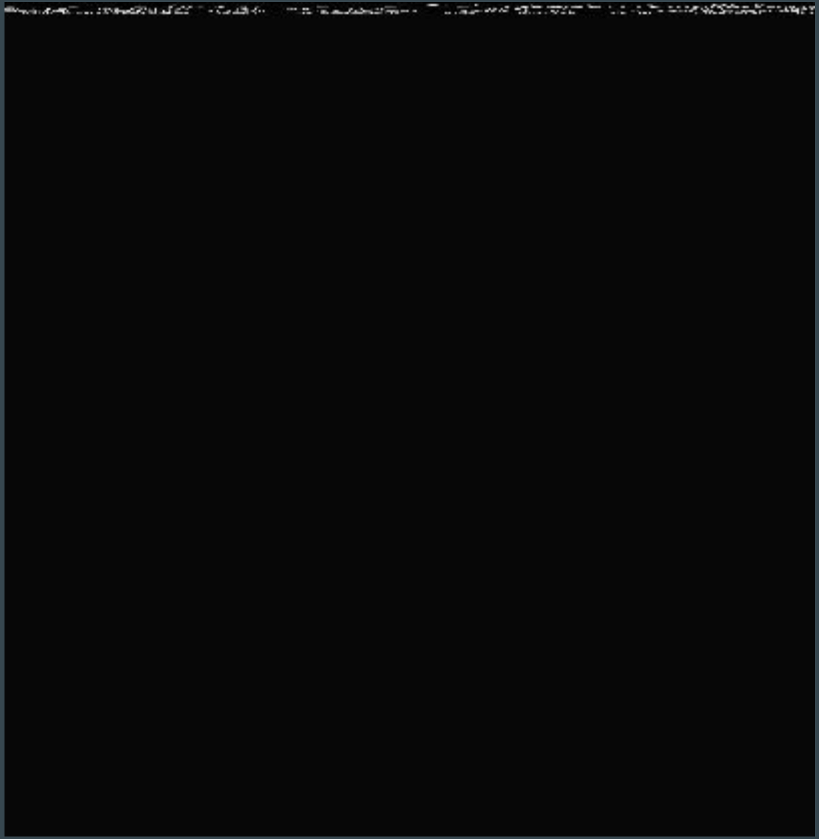


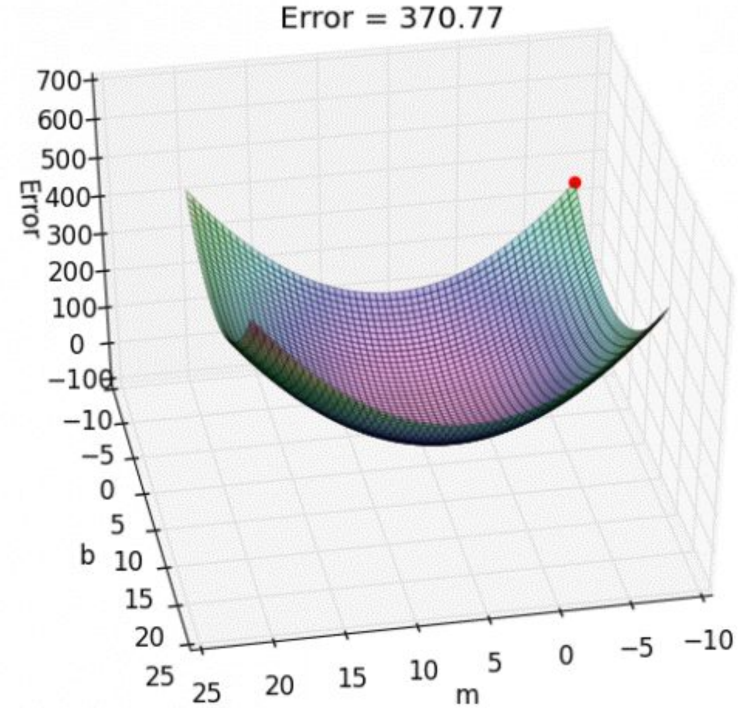
Choosing the right learning rate



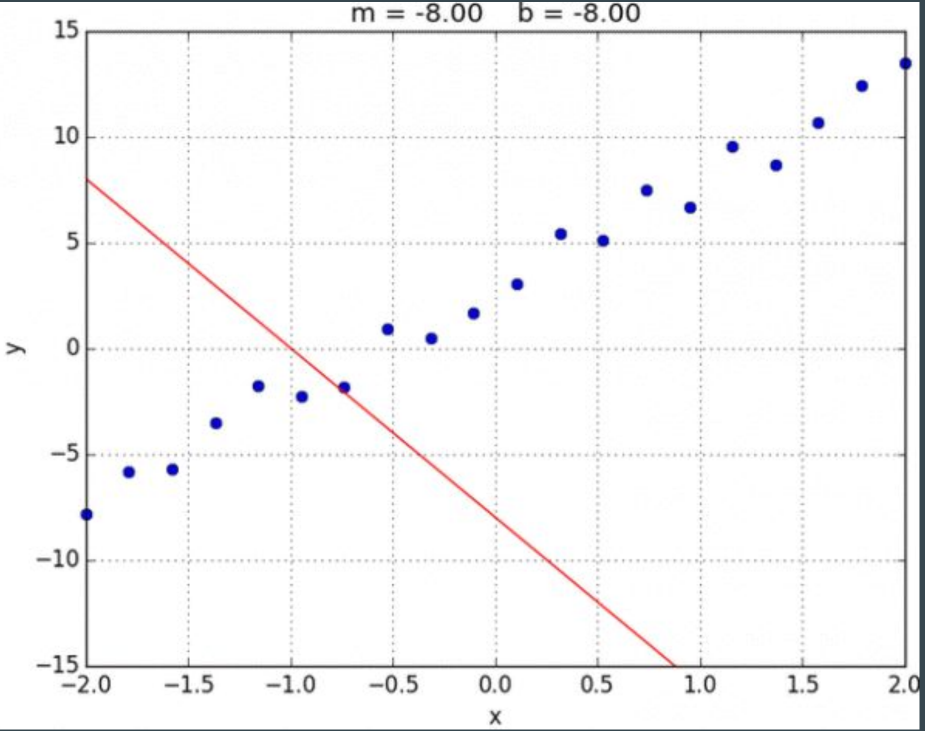
Gradient descent

- Randomize our weights
- Perturb our weights
- Calculate the gradient of our loss
- Update the weights
- Repeat for a number of epochs





Error = 370.77
Error
700
600
500
400
300
200
100
0
−100
−10
−5
0
5
b
10
15
20
25
25
20
15
10
5
0
−5
−10
m



m = -8.00   b = -8.00
15
10
5
0
−5
−10
−15
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x

Recap

Activation function ?

Loss function ?

Optimizer ?

Gradient descent ?

Epochs ?

Batch size ?

Learning Rate ?

Hyperparameters ?

Recap

Activation function - Makes our layer non-linear

Loss function - Tells us how far off our prediction was

Optimizer - Algorithm that defines the learning process

Gradient descent - One method used to update the training parameters

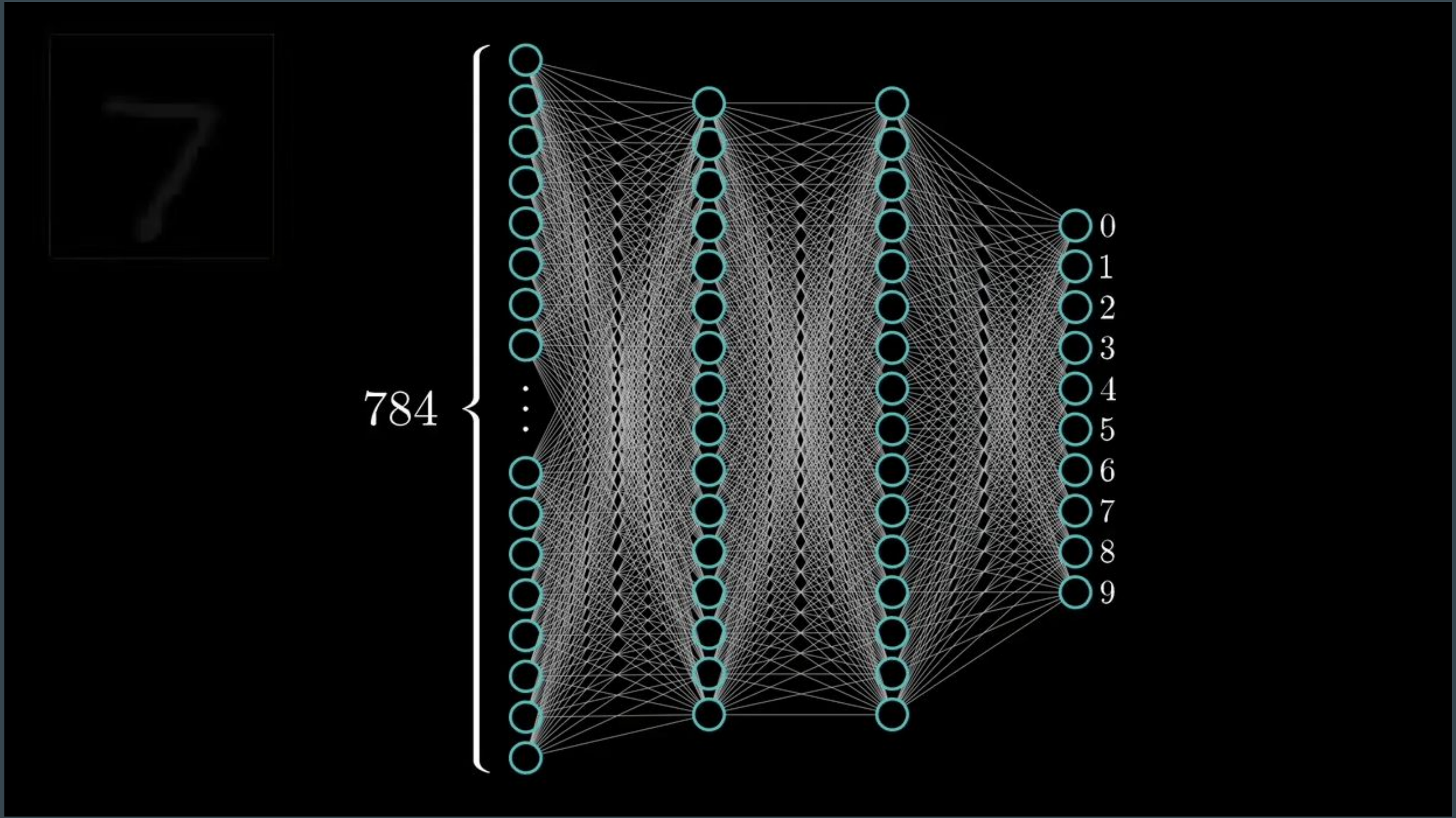
Epochs - The number of iterations we use to optimise the model

Batch size - The number of forward passes made on multiple data before updating the training parameters

Learning Rate - How great a step taken to update the training parameters

Hyperparameters - Non-trainable parameters

Questions ?



Web Links

- https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6
- http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html
- https://www.youtube.com/watch?v=aircAruvnKk
- https://www.coursera.org/specializations/deep-learning
- https://alykhantejani.github.io/images/