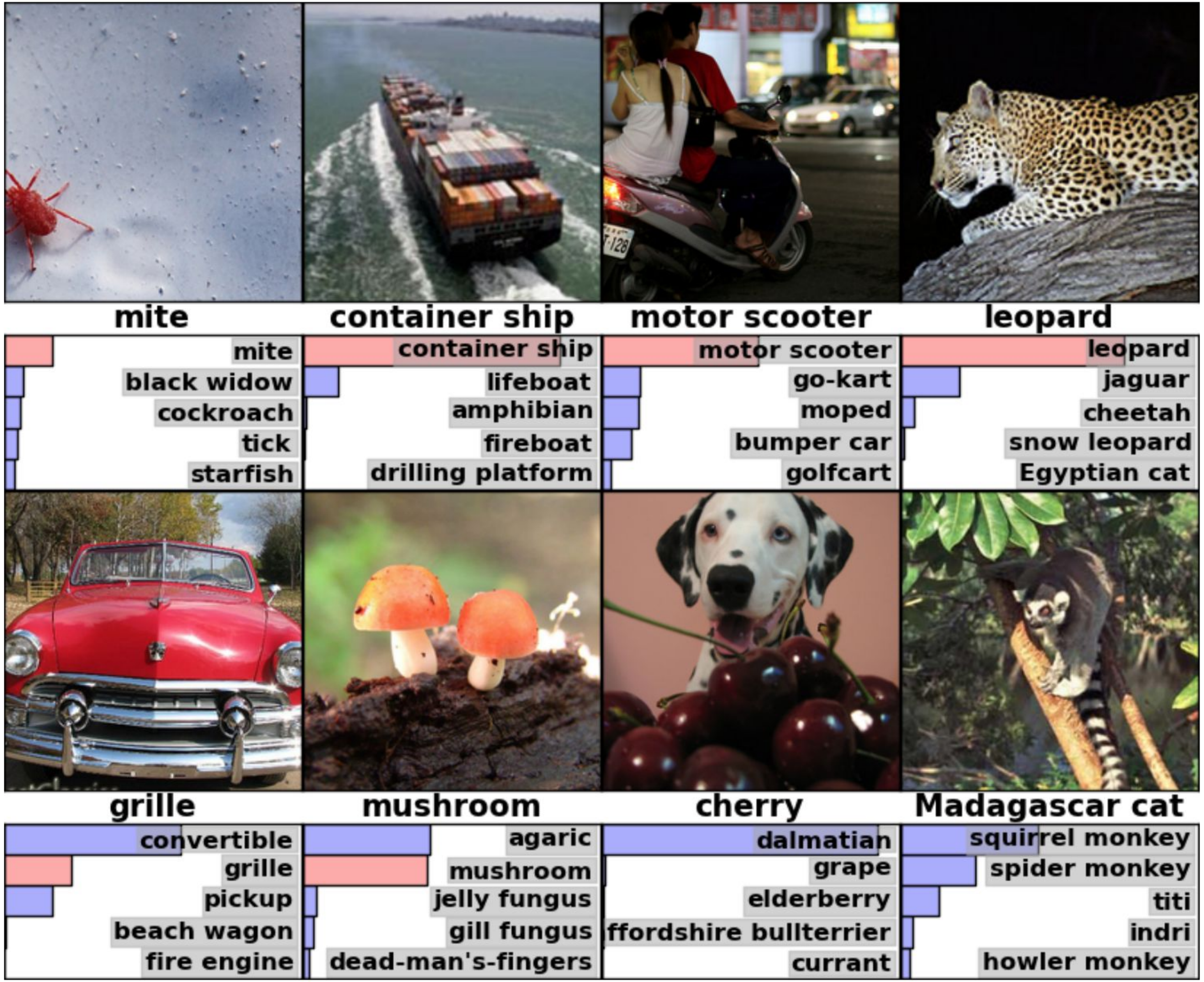
neural networks

an introduction

Slides contain adaptations from Geoffrey Hinton and Sebastian Raschka



mite
mite
black widow
cockroach
tick
starfish
container ship
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
motor scooter
go-kart
moped
bumper car
golfcart
leopard
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
convertible
grille
pickup
beach wagon
fire engine
mushroom
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
cherry
dalmatian
grape
elderberry
ffordshire bullterrier
currant
Madagascar cat
squirrel monkey
spider monkey
titi
indri
howler monkey

background

2012 - Geoffrey Hinton (University of Toronto)

Achieved 37.5% error (Top 1), 17% error (Top 5) on ImageNet database.

60 million parameters, 650,000 neurons

From then on: increased computational power helped with the popularity of neural nets (GPUs)

goal

- Motivate the structure
- What it means for a machine to learn
- What it means for learning to be ‘deep’
- Introduce these ideas using a vanilla neural network

approach

Classifier needs

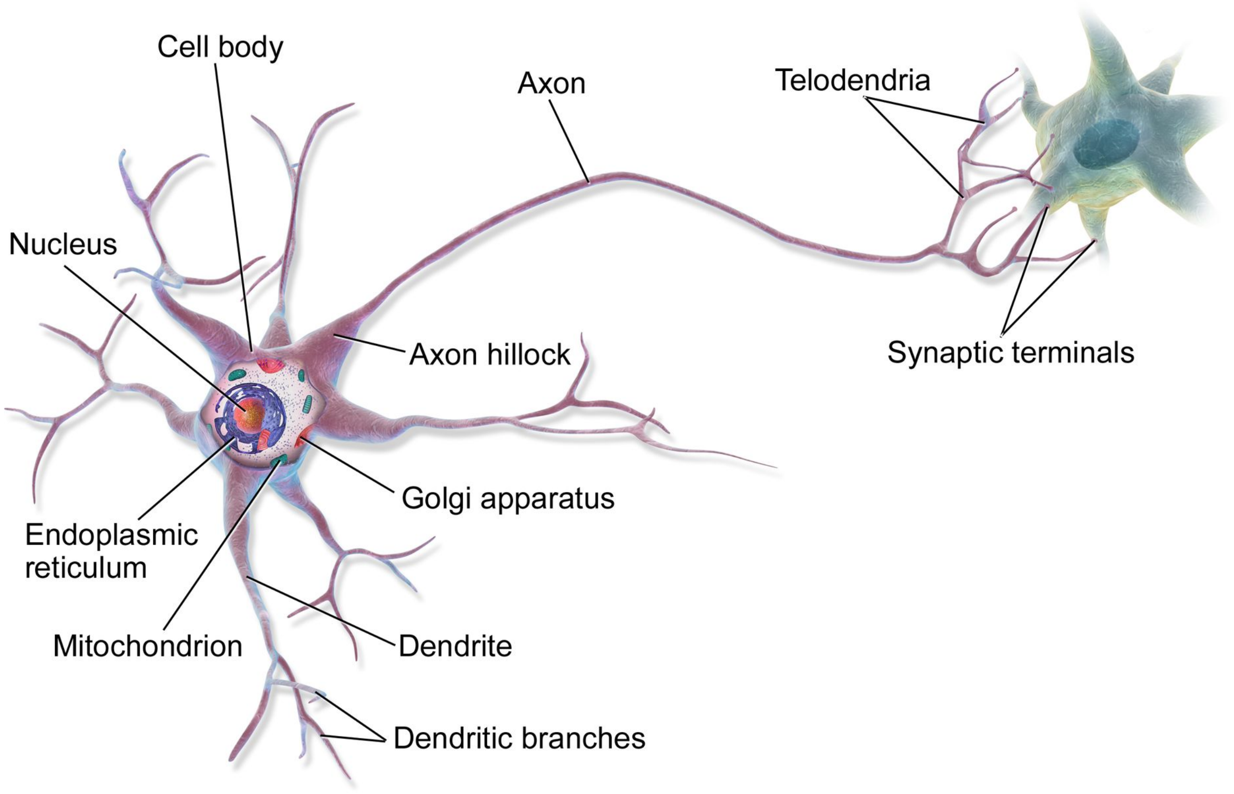
IT has to go through

then

Learning = Representation + Evaluation + Optimization

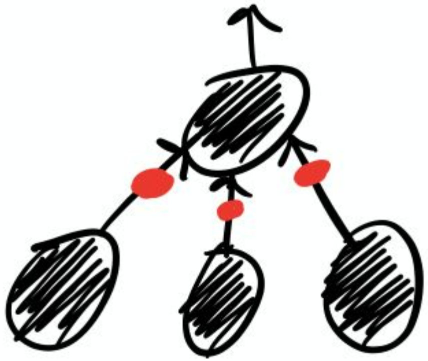
(Pedro Domingos, A Few Useful Things to Know about Machine Learning
https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf)

the inspiration



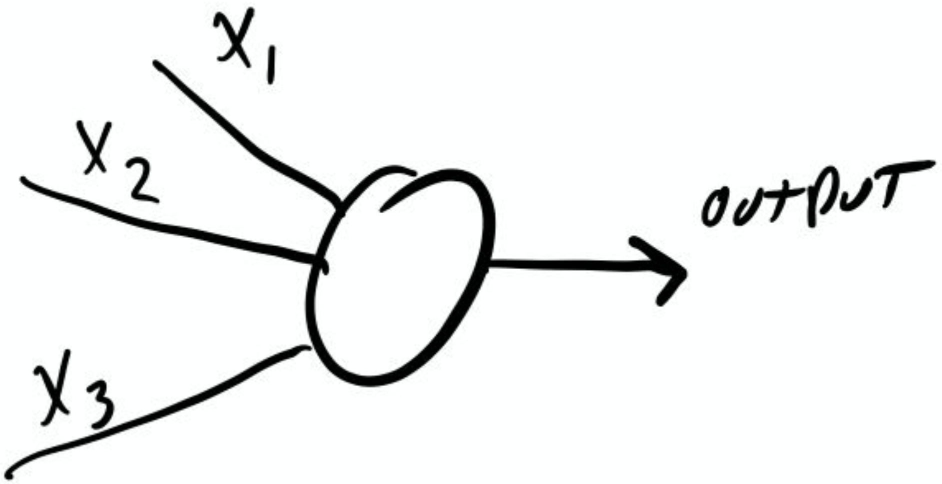
not a neuroscientist, but:

- Each neuron receives inputs from other neurons
- The effect of each input line on the neuron is controlled by a synaptic weight
  - positive or negative
- The synaptic weights adapt so that the whole network learns to perform useful computations
  - recognising objects, language, movement, etc.
- We have about 86 billion neurons

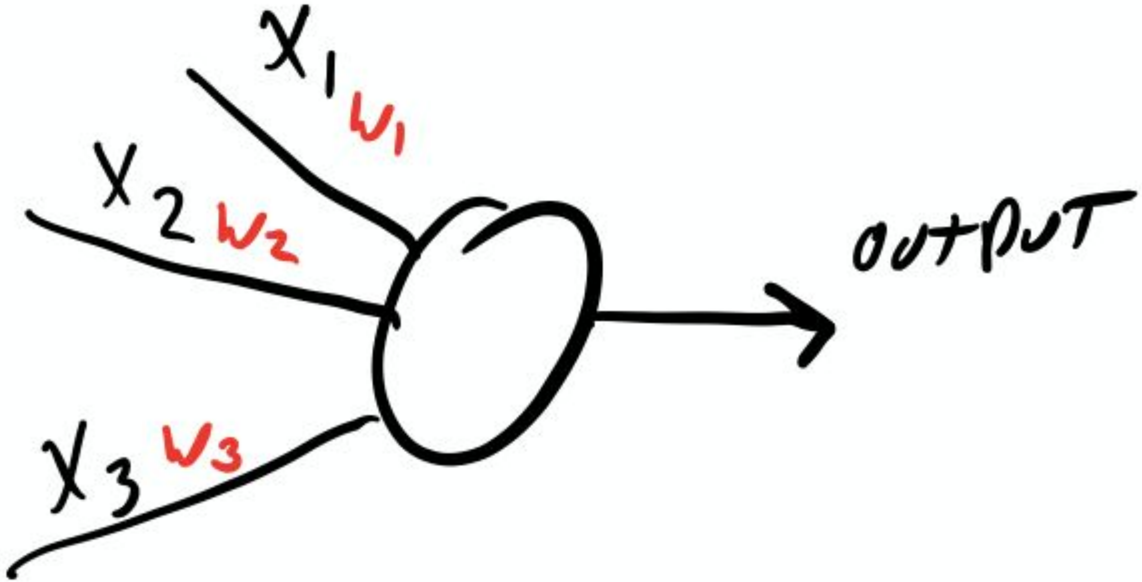


basic perceptron

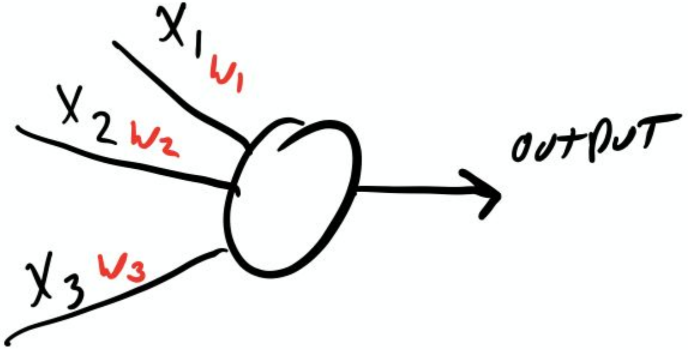
Type of artificial neuron



basic perceptron



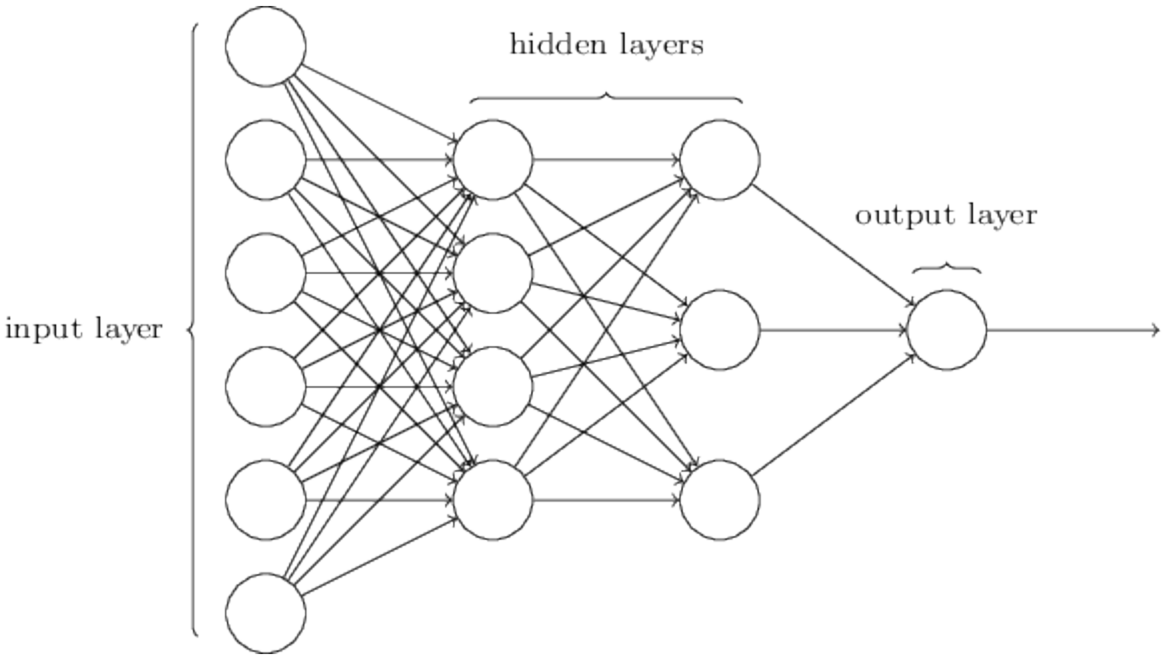
basic perceptron



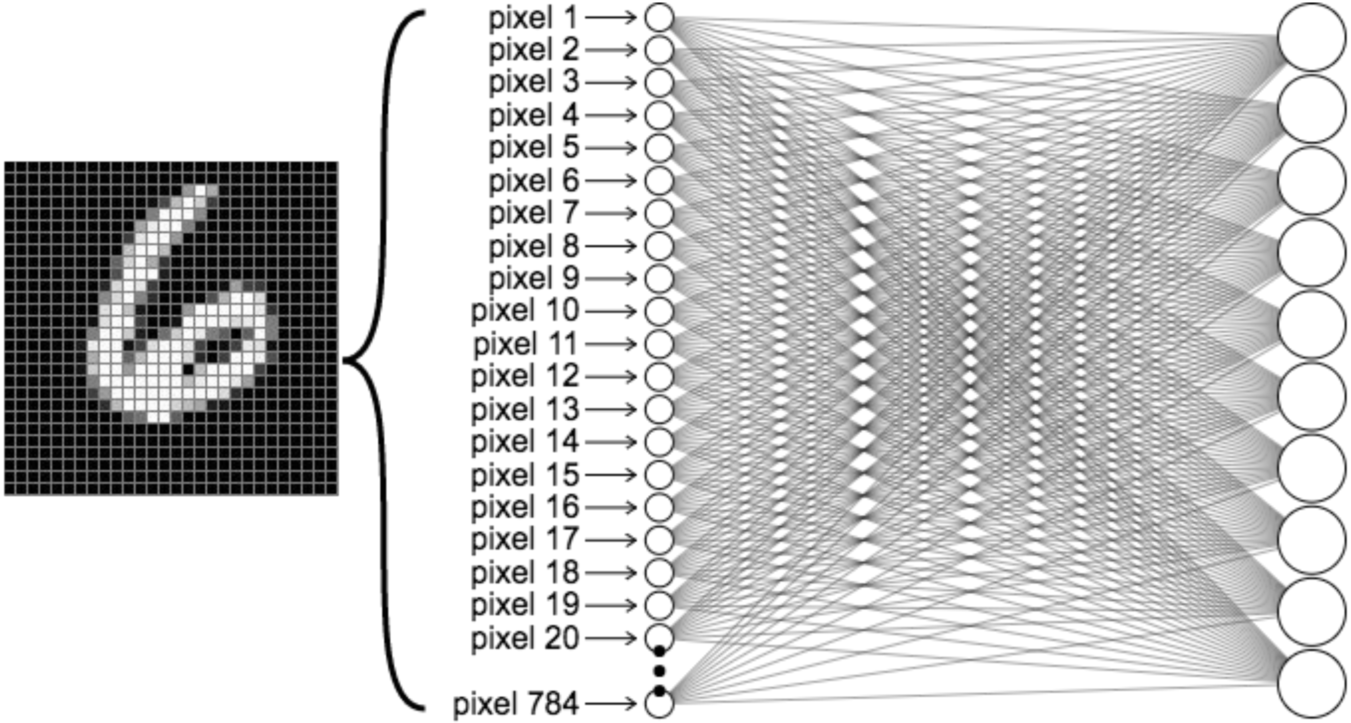
$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

combining

deep neural network simply means there are multiple hidden layers!

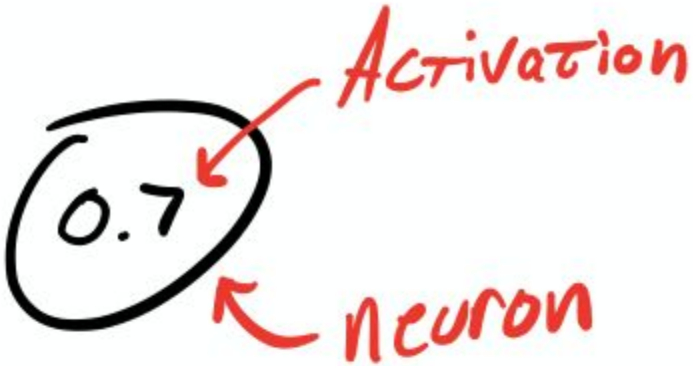


idea

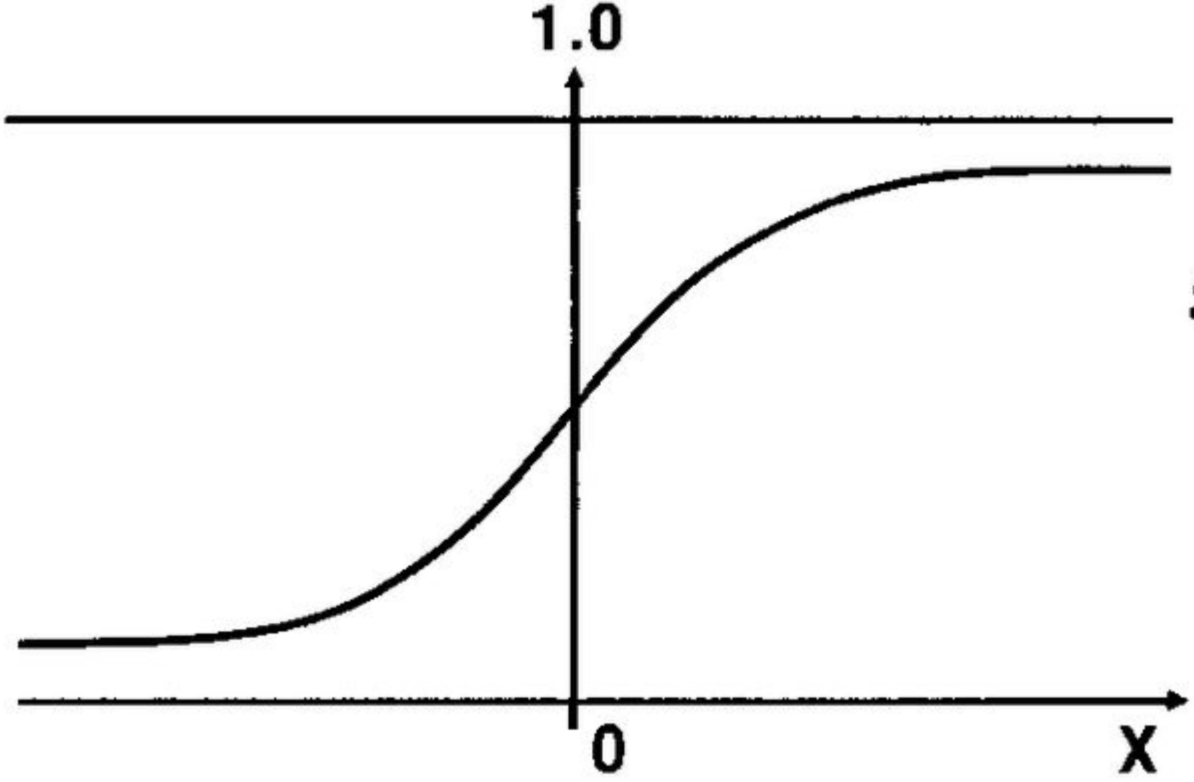

pixel 1
pixel 2
pixel 3
pixel 4
pixel 5
pixel 6
pixel 7
pixel 8
pixel 9
pixel 10
pixel 11
pixel 12
pixel 13
pixel 14
pixel 15
pixel 16
pixel 17
pixel 18
pixel 19
pixel 20
pixel 784

basic neuron

in our simplified model, a neuron simply holds a number



activation function: smooth



Sigmoid

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

learning

finding the right weights

$$\sigma(w_1 a_1 + w_2 a_2 + w_3 a_3 \cdots w_n a_n)$$

and biases

$$\sigma(w_1 a_1 + w_2 a_2 + w_3 a_3 \cdots w_n a_n \overset{\text{bias}}{-10})$$

what we have so far

- neurons
- layers of neurons with weights and biases
- activation function

what more do we need?

- some way to measure performance of our weights and biases
- some way to update our parameters based on how well they performed

measuring performance

adjust weights and threshold to maximise performance.

intuitively,

error = desired output − actual output

measuring performance

adjust weights and threshold to maximise performance.

intuitively,

error = desired output − actual output

error = || desired output − actual output ||

measuring performance

adjust weights and threshold to maximise performance.

intuitively,

$$\text{error} = \text{desired output} - \text{actual output}$$

$$\text{error} = \left|\text{desired output} - \text{actual output}\right|$$

$$\text{error} = (\text{desired output} - \text{actual output})^2$$

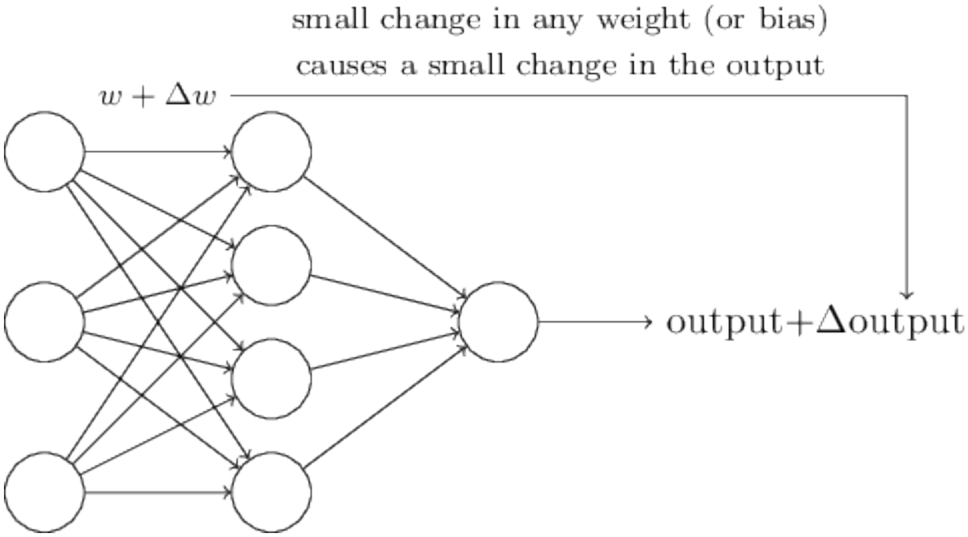
measuring performance

Input	actual	Desired	Absolute Error	Square Error
0	0	0	0	0
1	3	2	1	1
2	6	4	2	4
3	9	6	3	9
4	12	8	4	16
Total:	-	-	**10**	**30**

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 .$$

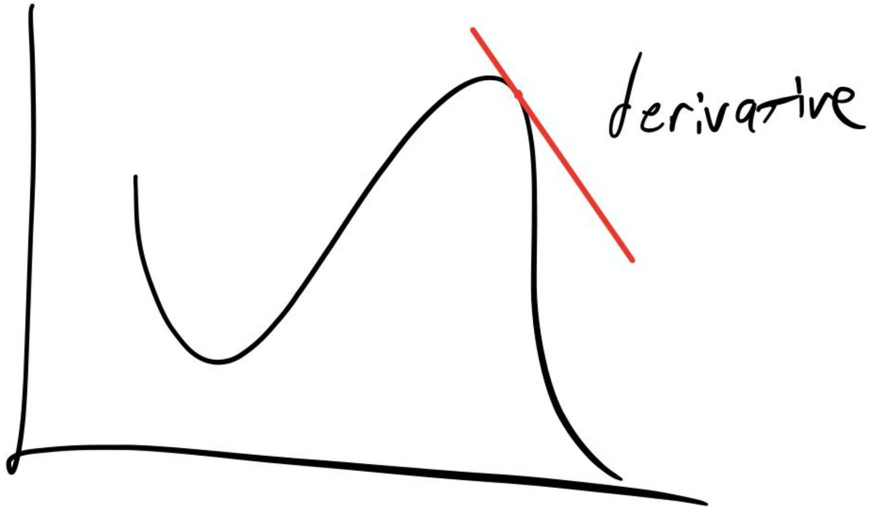
gradient descent

- now that we have the error, what we want to do is find the combination of weights and biases that minimise it



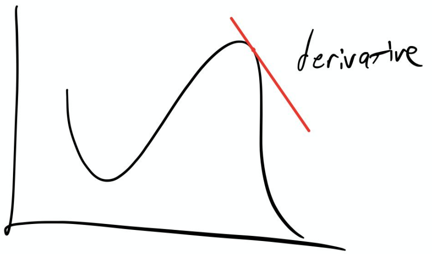
gradient descent

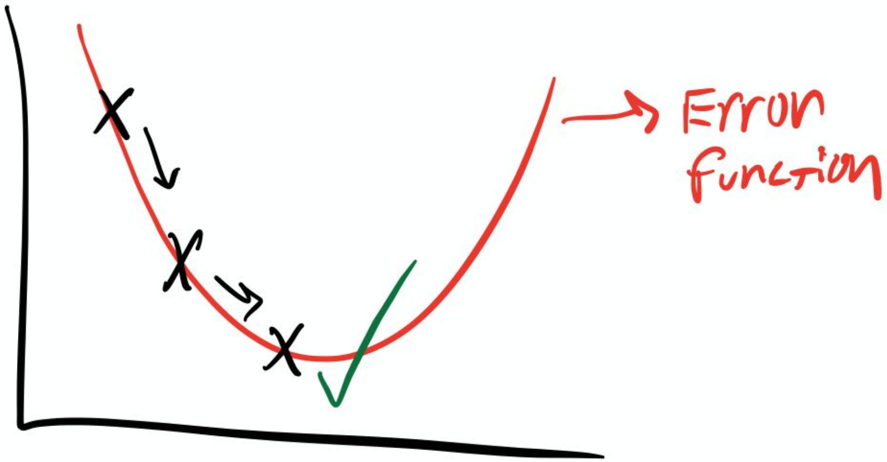
- now that we have the error, what we want to do is find the combination of weights and biases that minimise it



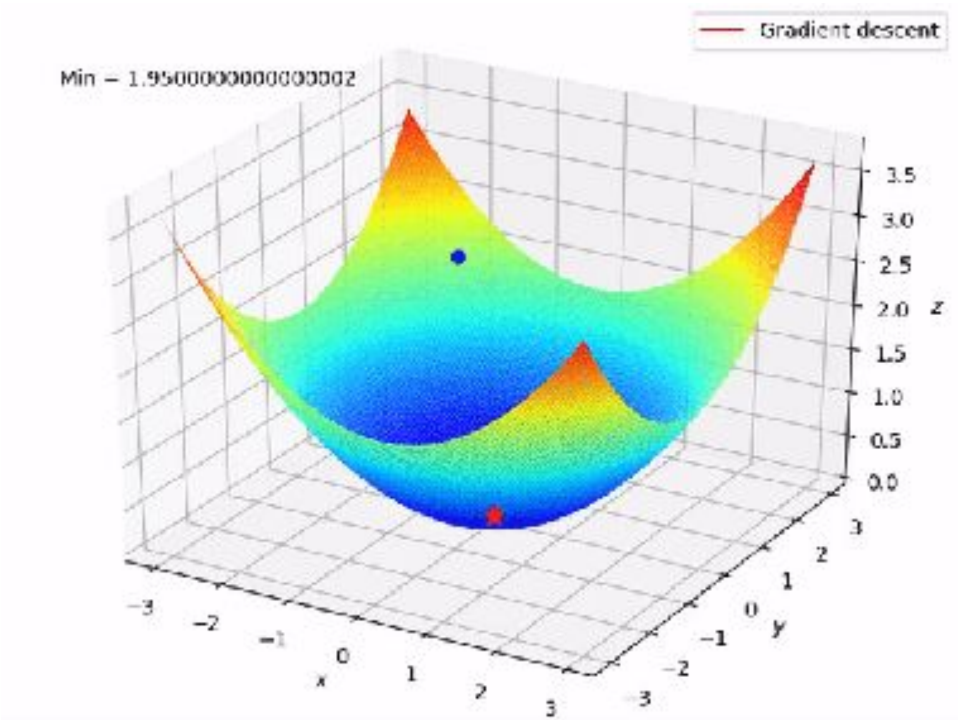
gradient descent

- now that we have the error, what we want to do is find the combination of weights and biases that minimise it



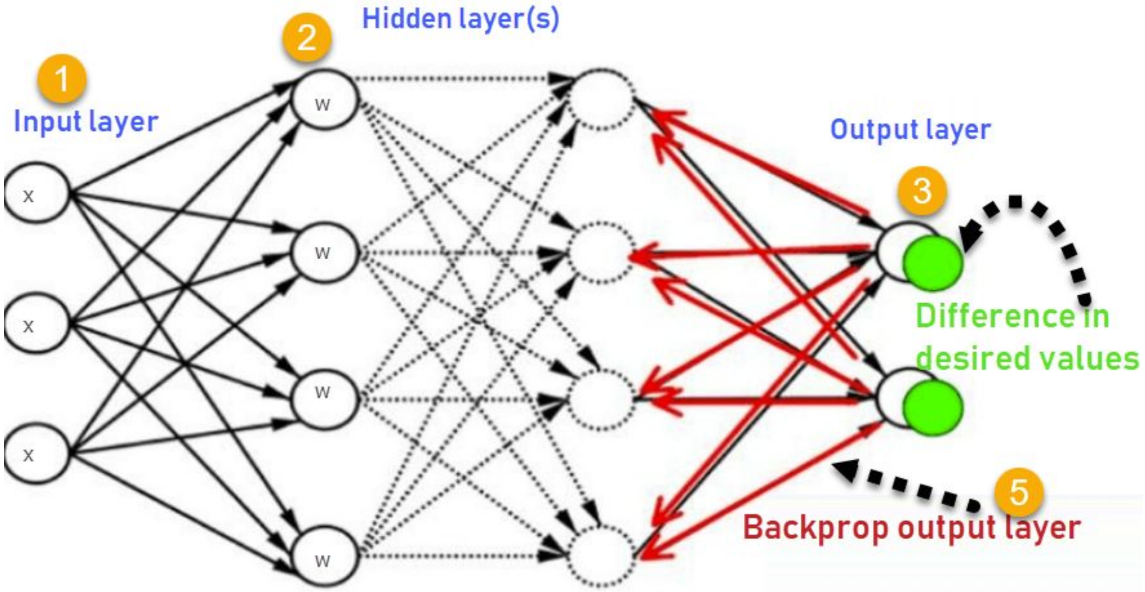


gradient descent



backpropagation

- gradient descent is part of a process called backpropagation
- first we calculate for each neuron how much we want to change, and then send this backwards



what do we have now

recall: Learning = Representation + Evaluation + Optimization

- representation: neural network
- evaluation: mean squared error
- optimization: gradient descent + backpropagation

tensorflow playground

neural networks in R

‘neuralnet’ package