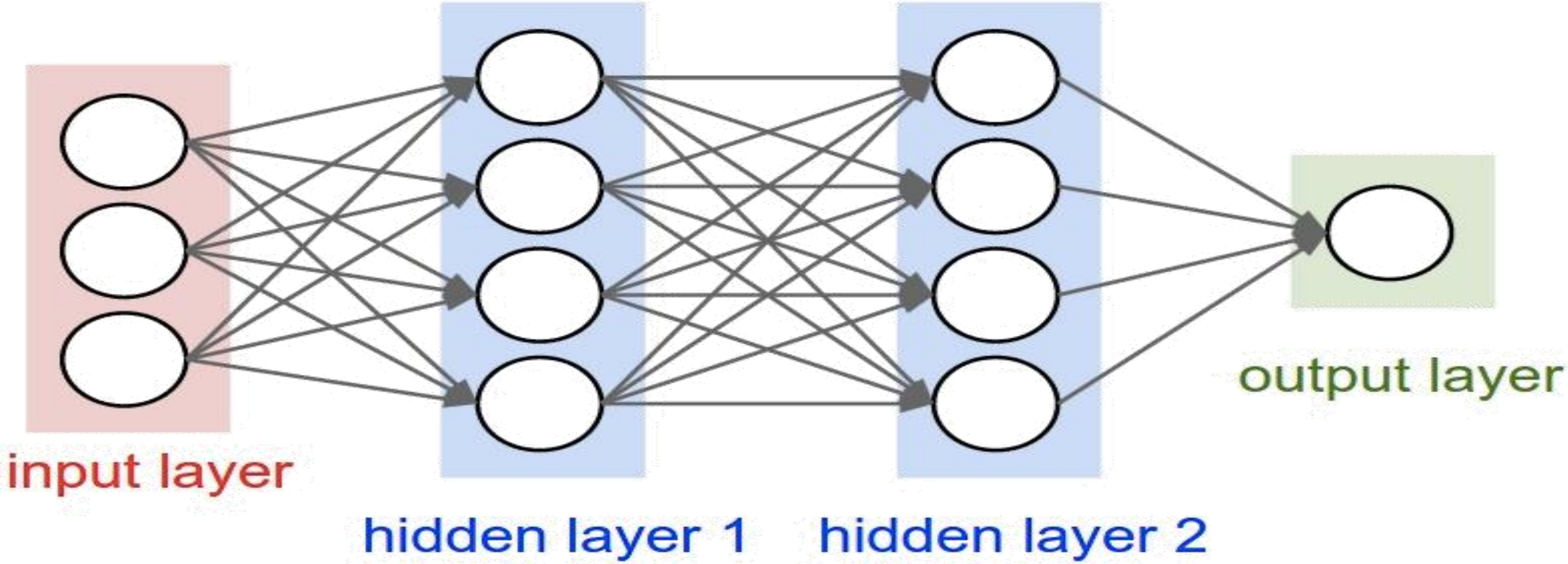
Neural Networks

Introduction

- Rebirth of neural networks
- Inspired by the human brain (networks of neurons)



History of Deep Learning

Golden Age (1960–1969) · "Dark Age" (1969–1986) · "Dark Age" (2000–2006) · Golden Age (2006–)

- Electronic Brain 1943
- Perceptron 1957
- Delta Learning Rule 1960
- XOR Problem 1969
- Neocognitron 1980
- Backpropagation 1986
- SVMs 1996
- Convolutional Neural Nets 1998
- Hierarchical Feature Learning 2006
- AlexNet 2012

1940 | 1950 | 1960 | 1970 | 1980 | 1990 | 2000 | 2010


McCulloch-Pitts


Rosenblatt


Widrow-Hoff

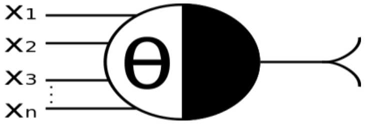

Minsky-Papert

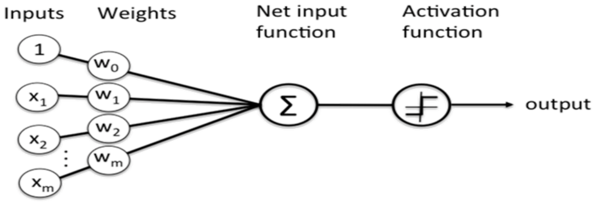

Rumelhart-Hinton-Williams

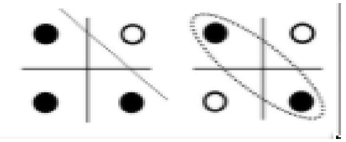

LeCun

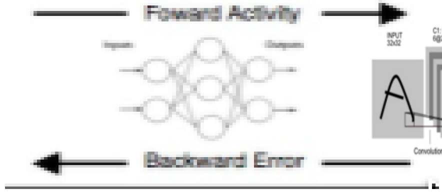

Hinton-Ruslan

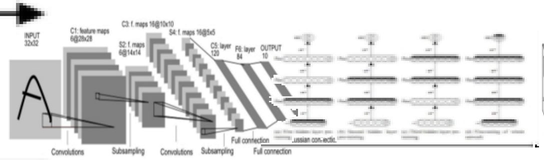

Krizhevsky-Sutskever-Hinton

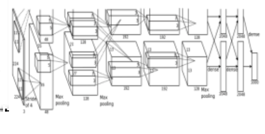




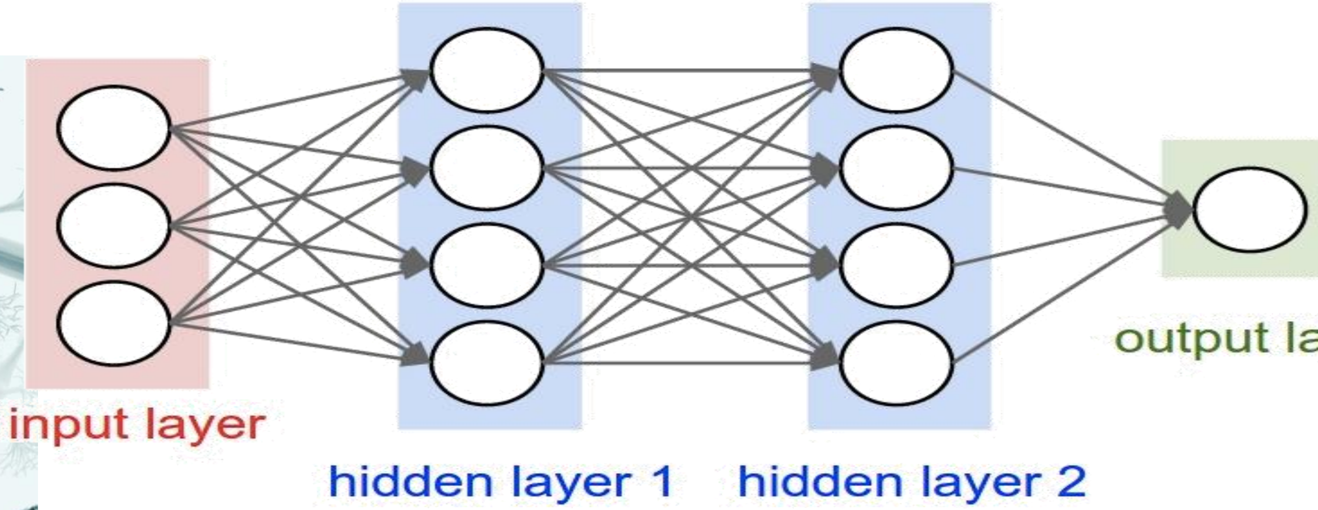






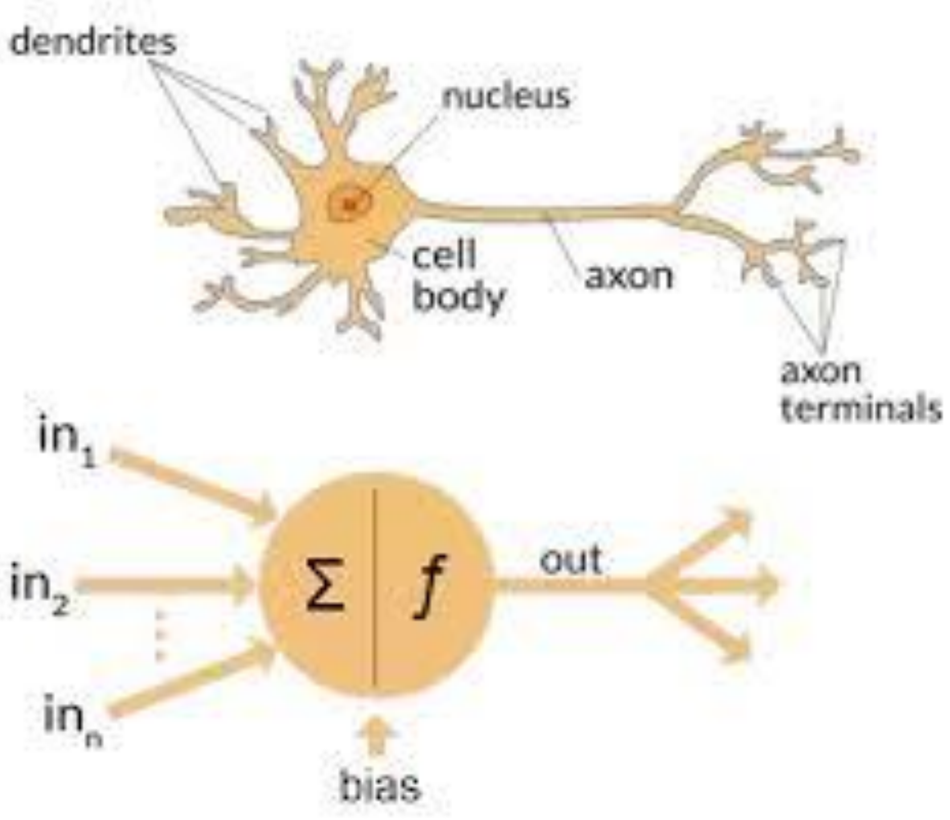


Neural Networks



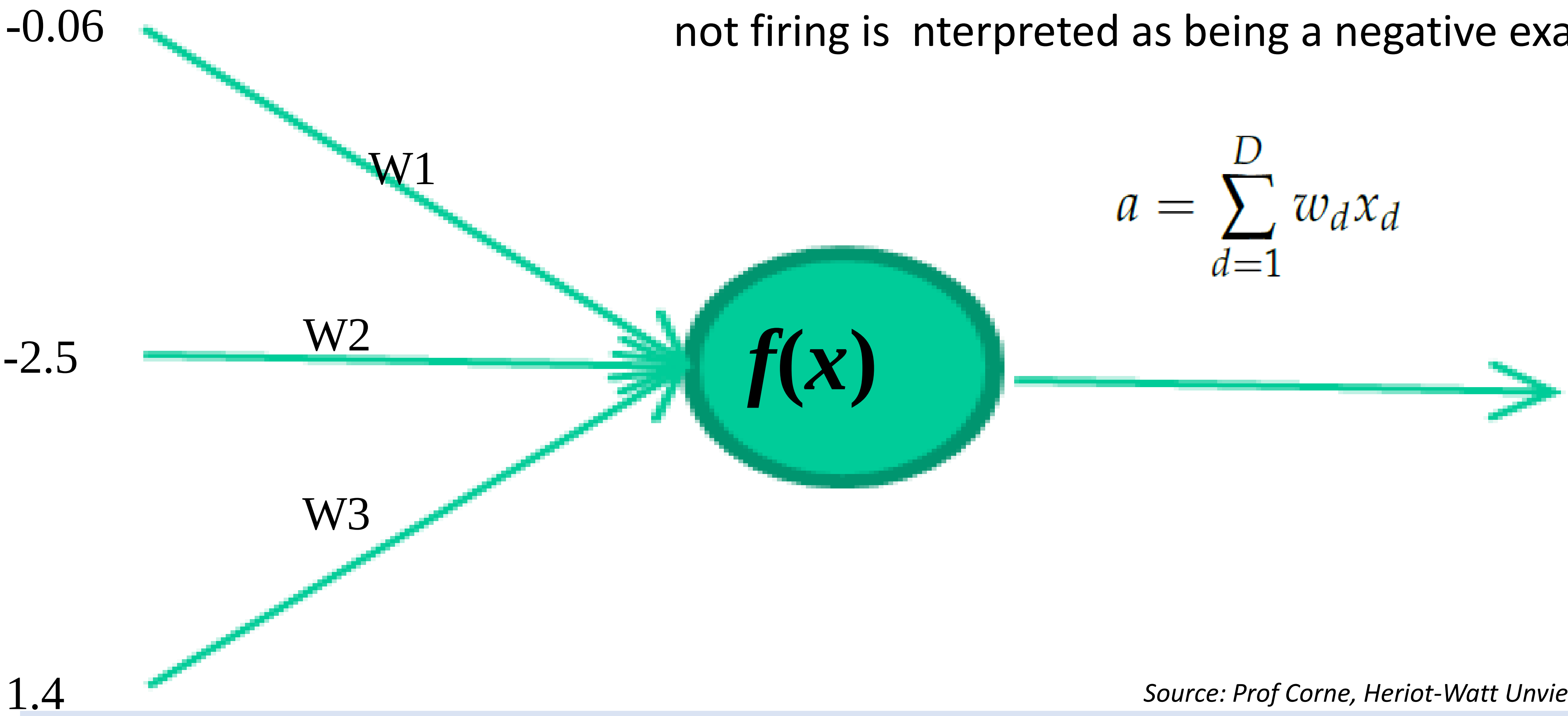
Neuron

- Based on how much these incoming neurons are firing, and how “strong” the neural connections are, our main neuron will “decide” how strongly it wants to fire.
- Learning in the brain happens by neurons becoming connected to other neurons, and the strengths of connections adapting over time.
- Receives input from D-many other neurons, one for each input feature. The strength of these inputs are the feature values.



Neural networks

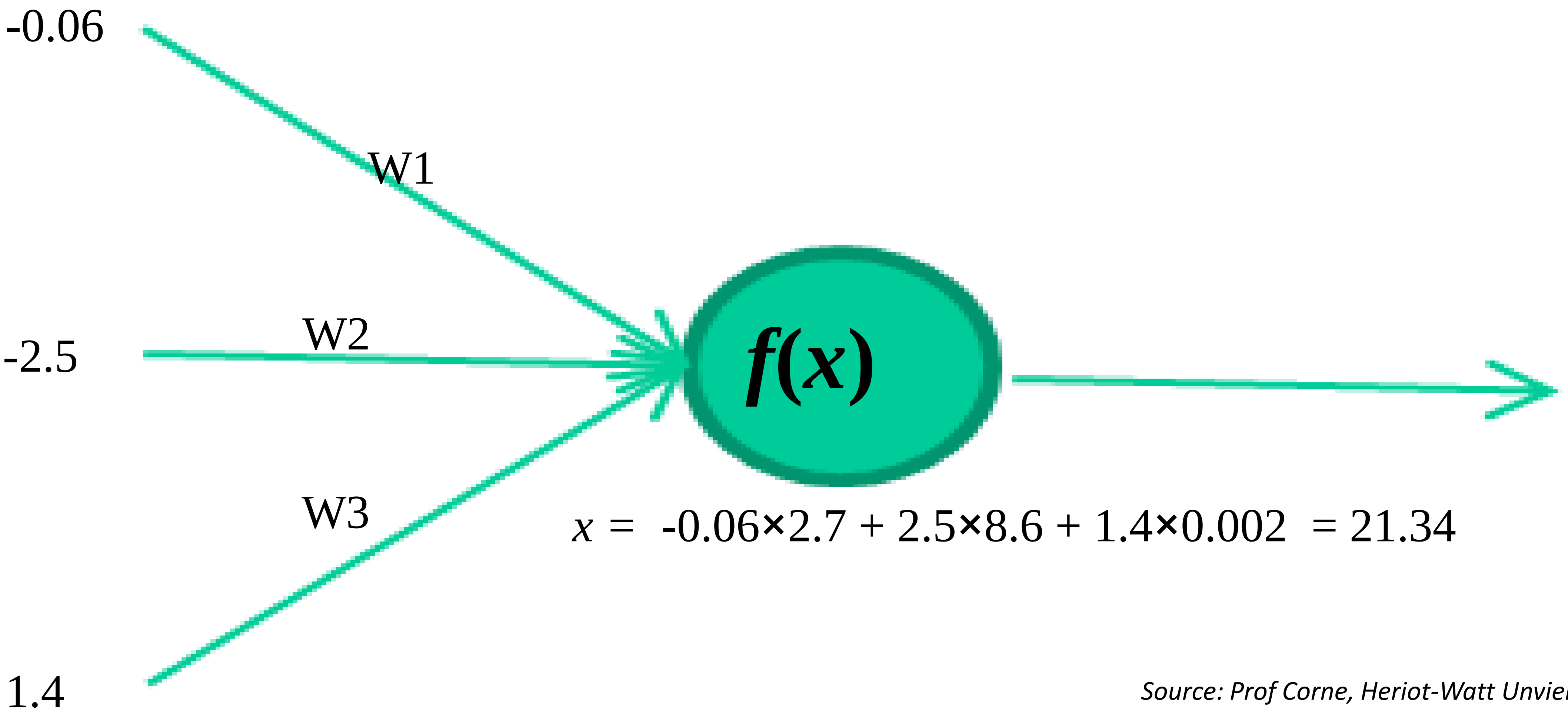
Firing is interpreted as being a positive example and not firing is nterpreted as being a negative example



$$a = \sum_{d=1}^{D} w_d x_d$$

Source: Prof Corne, Heriot-Watt Unviersity, UK

Deep Learning

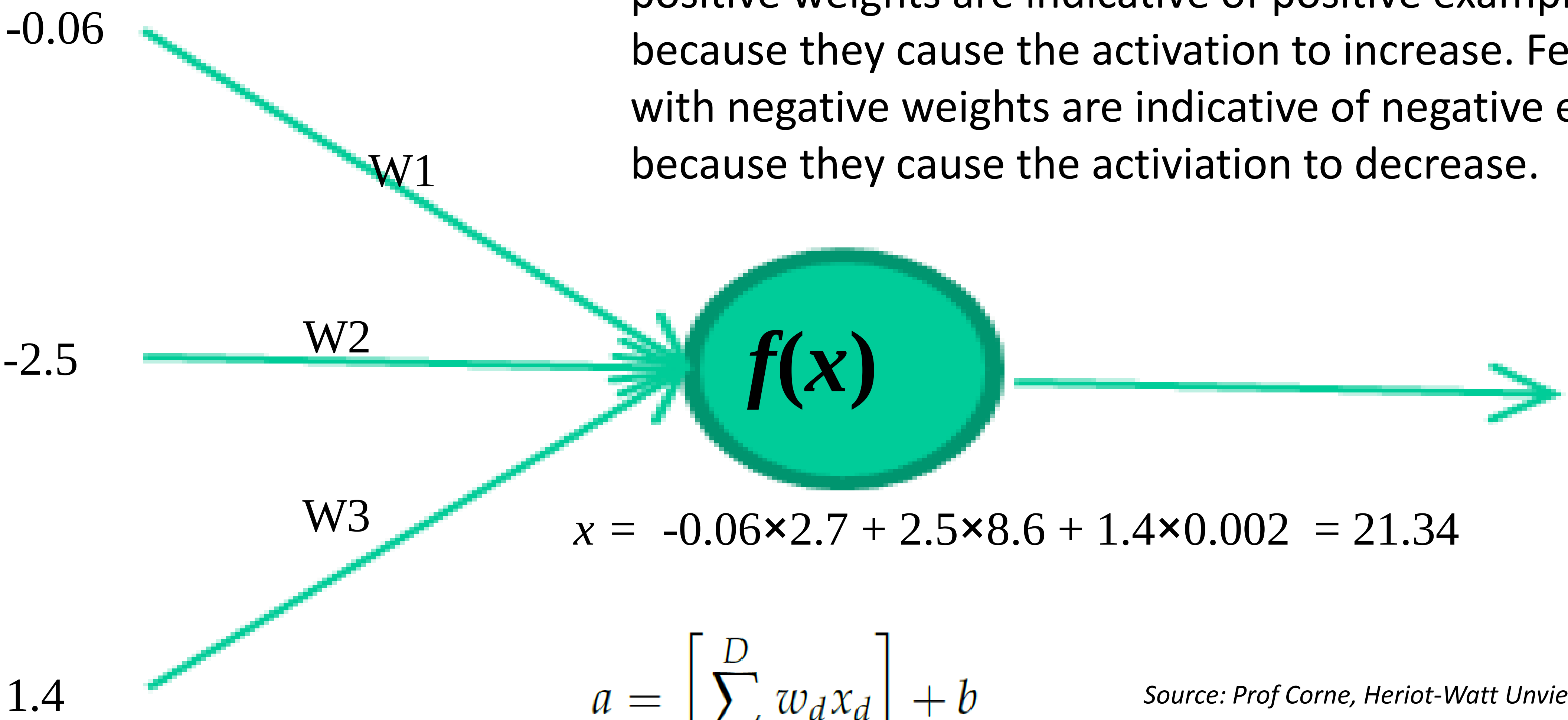


$x = -0.06\times2.7 + 2.5\times8.6 + 1.4\times0.002 = 21.34$

Source: Prof Corne, Heriot-Watt Unviersity, UK

Deep Learning

Features with zero weight are ignored. Features with positive weights are indicative of positive examples because they cause the activation to increase. Features with negative weights are indicative of negative examples because they cause the activiation to decrease.



$$x = -0.06\times 2.7 + 2.5\times 8.6 + 1.4\times 0.002 = 21.34$$

$$a = \left[\sum_{d=1}^{D} w_d x_d\right] + b$$

Source: Prof Corne, Heriot-Watt Unviersity, UK

Perceptron Algorithm

- **Online**. This means hat instead of considering the entire data set at the same time, it only ever looks at one example.
- **error driven.** This means that, so long as it is doing well, it doesn’t bother updating its parameters

Perceptron Algorithm

Algorithm 5 PerceptronTrain($\mathbf{D}$, $MaxIter$)

$w_d \leftarrow 0$, for all $d = 1 \dots D$ // initialize weights
$b \leftarrow 0$ // initialize bias
for $iter = 1 \dots MaxIter$ **do**
  for all $(x,y) \in \mathbf{D}$ **do**
    $a \leftarrow \sum_{d=1}^{D} w_d \, x_d + b$ // compute activation for this example
    if $ya \leq 0$ **then**
      $w_d \leftarrow w_d + yx_d$, for all $d = 1 \dots D$ // update weights
      $b \leftarrow b + y$ // update bias
    end if
  end for
end for
return $w_0, w_1, \dots, w_D, b$

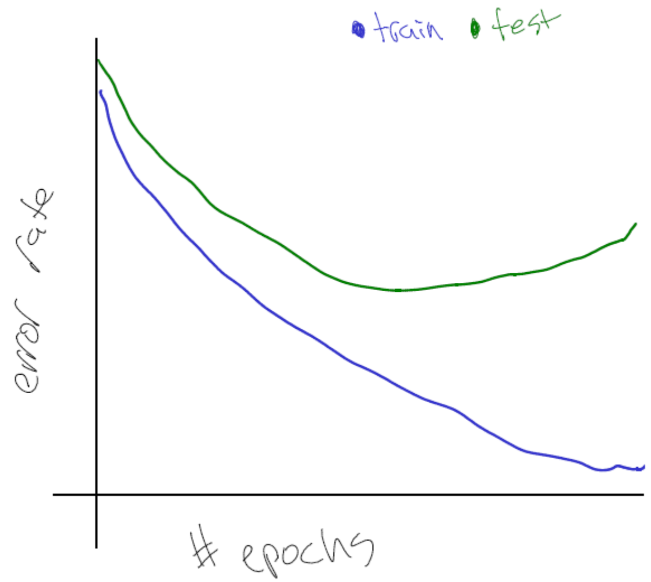
Perceptron algorithm

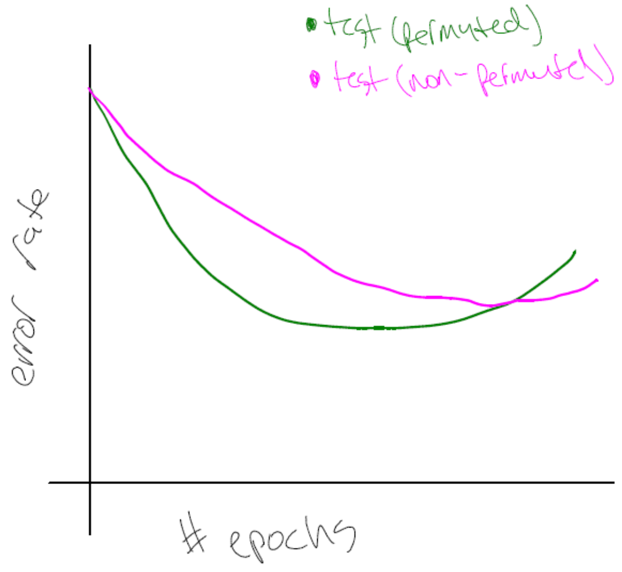
- weight wd is increased by yxd and the bias is increased by y.
- goal of the update is to adjust the parameters so that they are "better' for the current example

$$
\begin{aligned}
a' &= \sum_{d=1}^{D} w'_d x_d + b' \\
&= \sum_{d=1}^{D} (w_d + x_d) x_d + (b + 1) \\
&= \sum_{d=1}^{D} w_d x_d + b + \sum_{d=1}^{D} x_d x_d + 1 \\
&= a + \sum_{d=1}^{D} x_d^2 + 1 \quad > \quad a
\end{aligned}
$$

Perceptron algorithm

- If we make many many passes over the training data, then the algorithm is likely to overfit. On the other hand, going over the data only one time might lead to underfitting
- loop over all the training examples in a constant
- Order is a bad idea. re-permute the examples in each iteration.

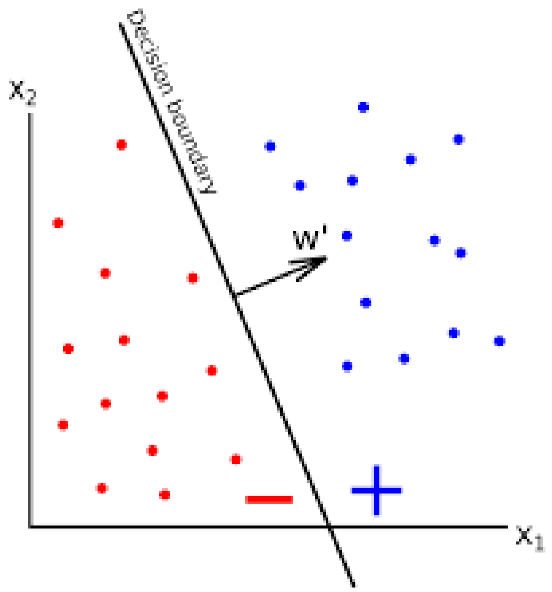


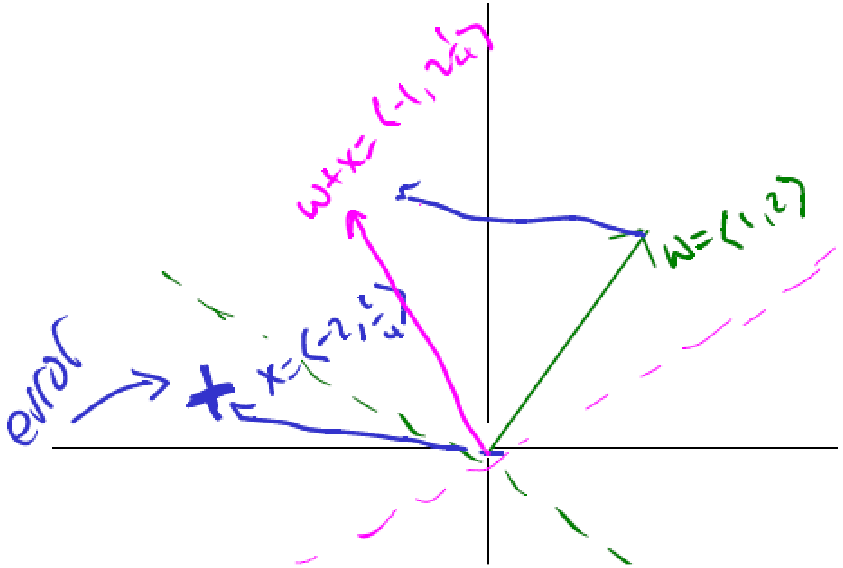


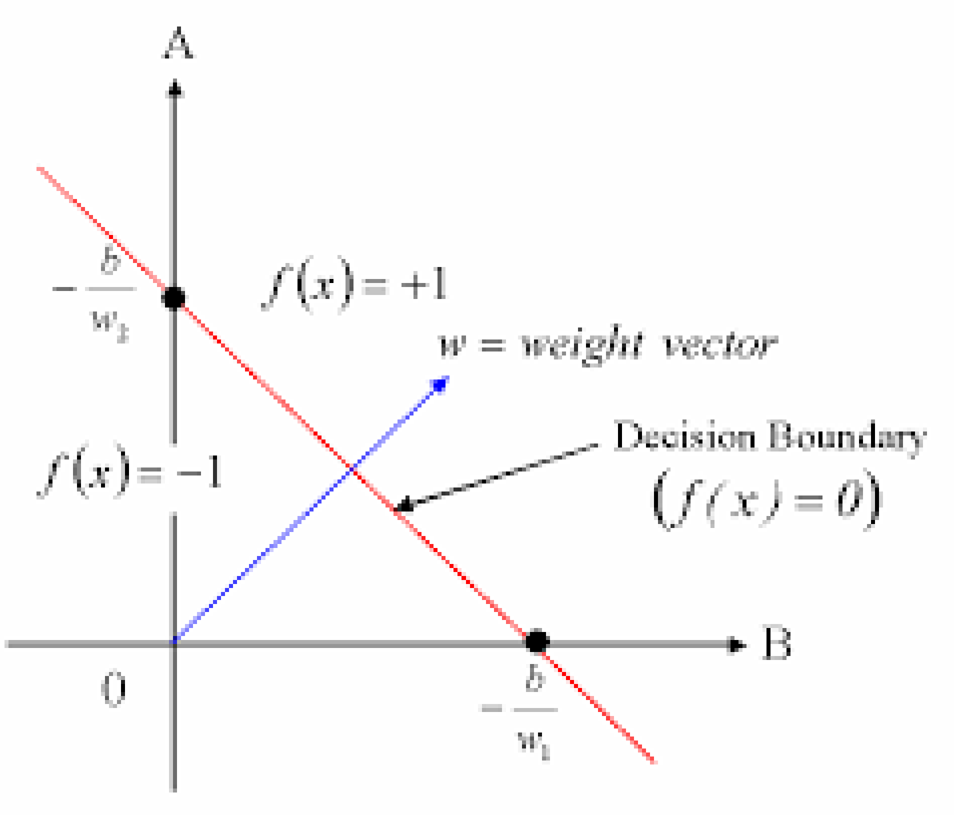
Perceptron decision boundary

$$\mathcal{B} = \left\{ x \; : \; \sum_d w_d x_d = 0 \right\}$$

normalized weight vectors, **w x** is just the distance of **x, w** is exactly the activiation of that example







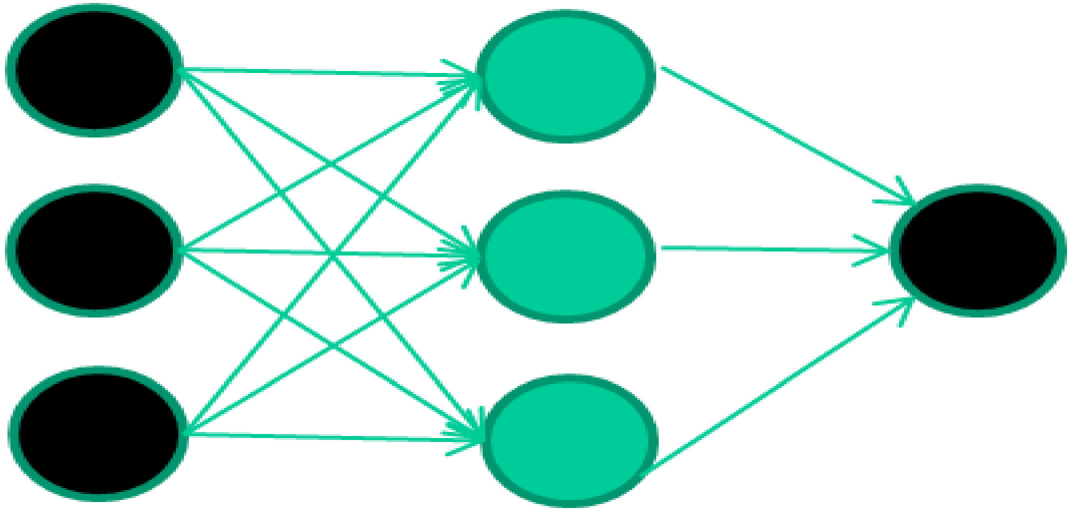
Deep Learning

How do they learn?

A dataset

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …



Source: Prof Corne, Heriot-Watt Unviersity, UK

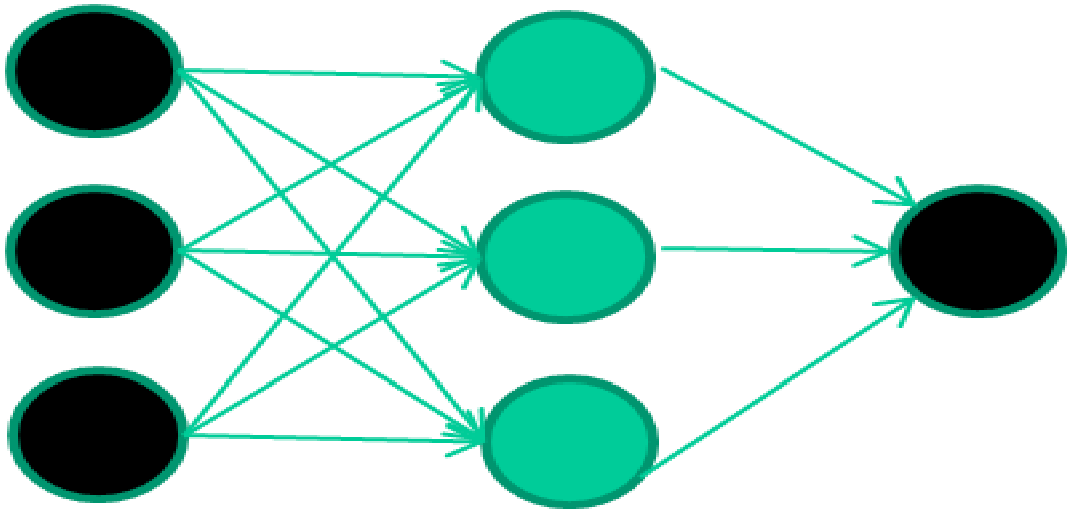
Deep Learning

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Initialise with random weights



Source: Prof Corne, Heriot-Watt Unviersity, UK

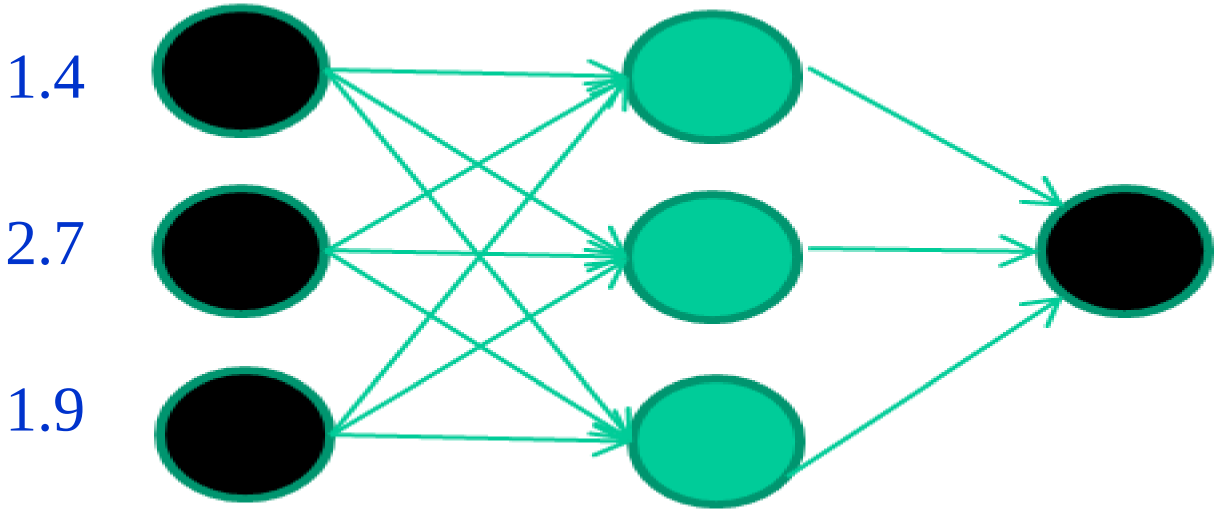
Deep Learning

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Present a training pattern



Source: Prof Corne, Heriot-Watt Unviersity, UK

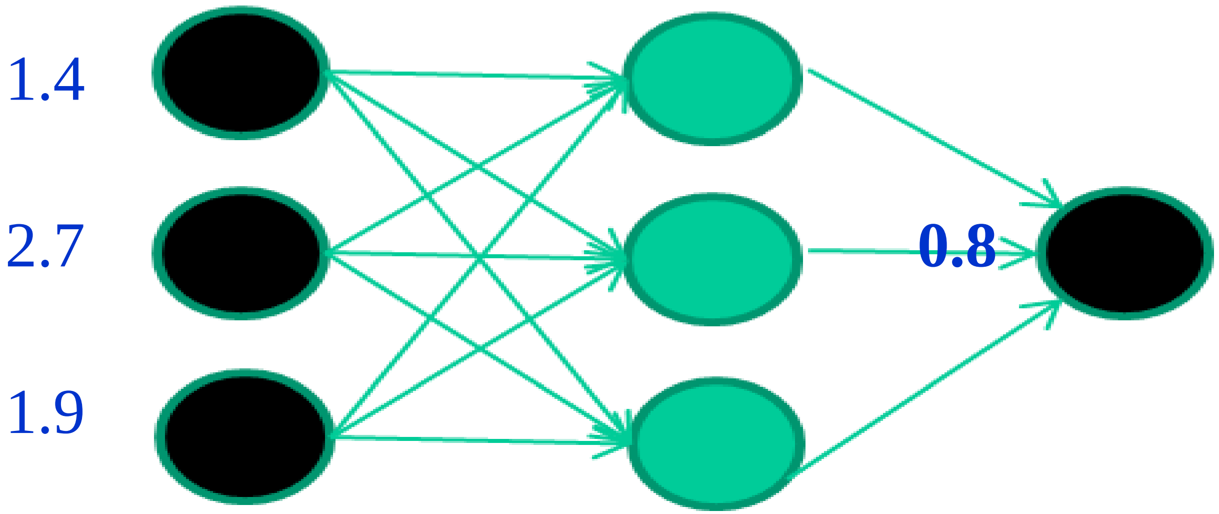
Deep Learning

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Feed it through to get output



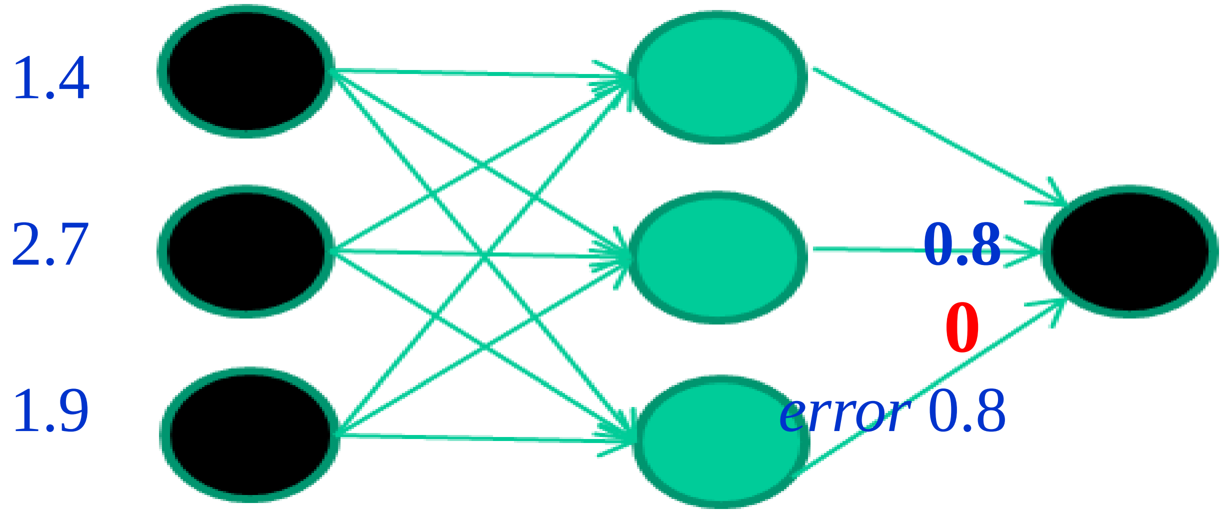
Source: Prof Corne, Heriot-Watt Unviersity, UK

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Compare with target output



Source: Prof Corne, Heriot-Watt Unviersity, UK

Deep Learning

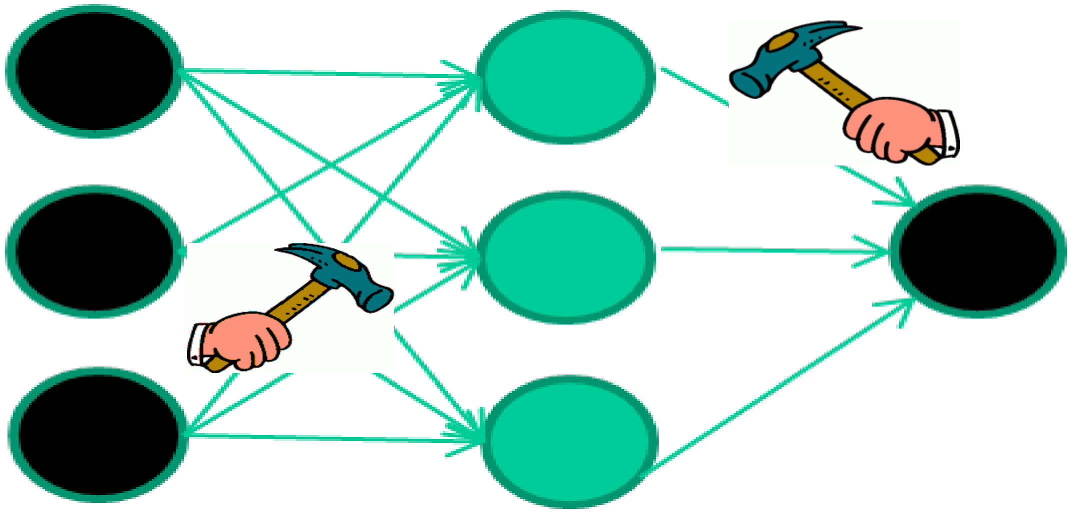
How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

Adjust weights based on error



Source: Prof Corne, Heriot-Watt Unviersity, UK

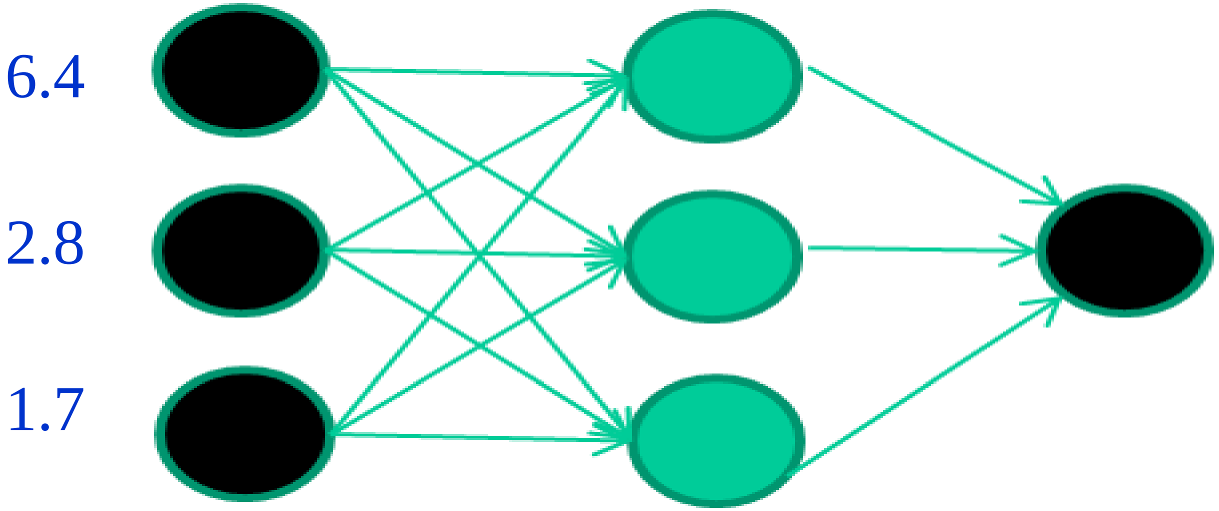
Deep Learning

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Present a training pattern



Source: Prof Corne, Heriot-Watt Unviersity, UK

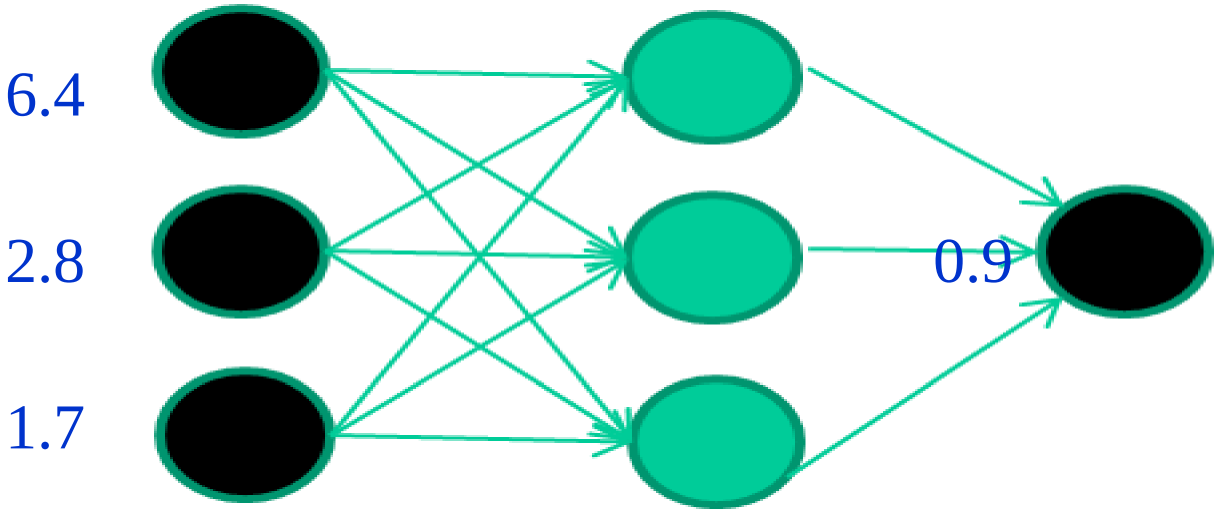
Deep Learning

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Feed it through to get output



Source: Prof Corne, Heriot-Watt Unviersity, UK

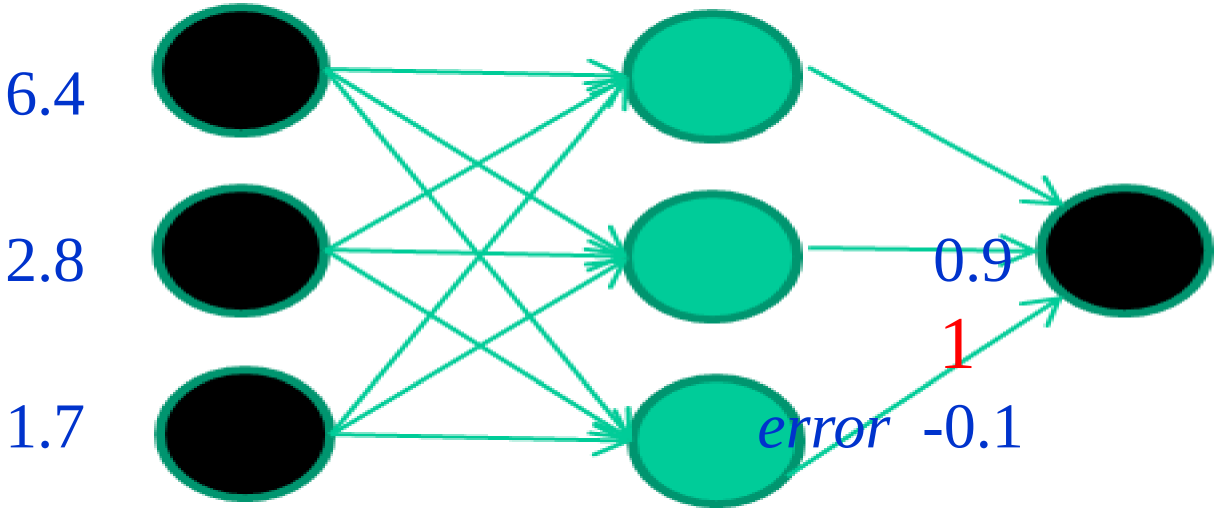
Deep Learning

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Compare with target output



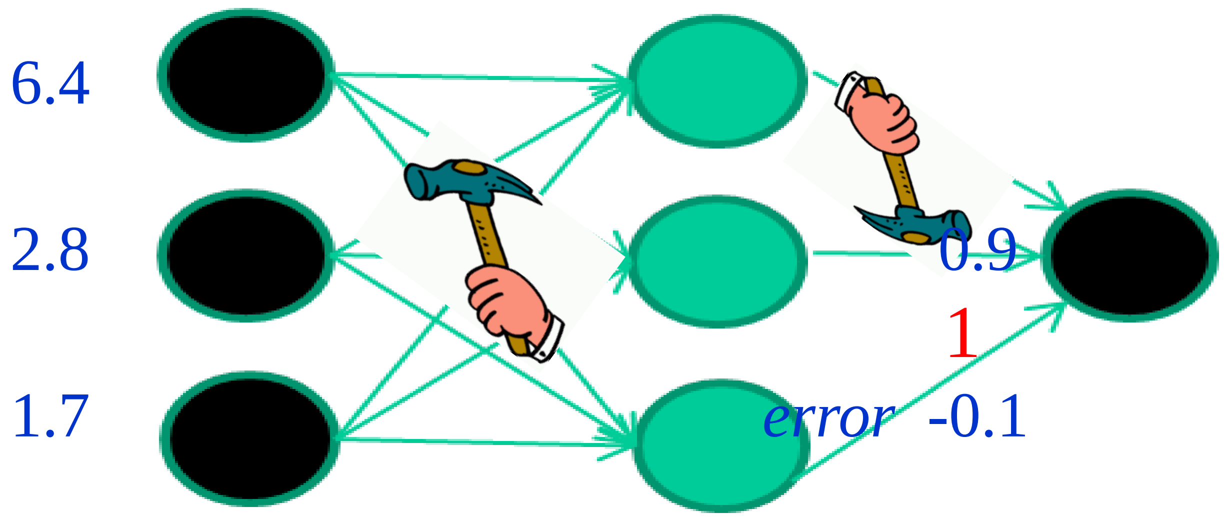
Source: Prof Corne, Heriot-Watt Unviersity, UK

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Adjust weights based on error



Source: Prof Corne, Heriot-Watt Unviersity, UK

Deep Learning

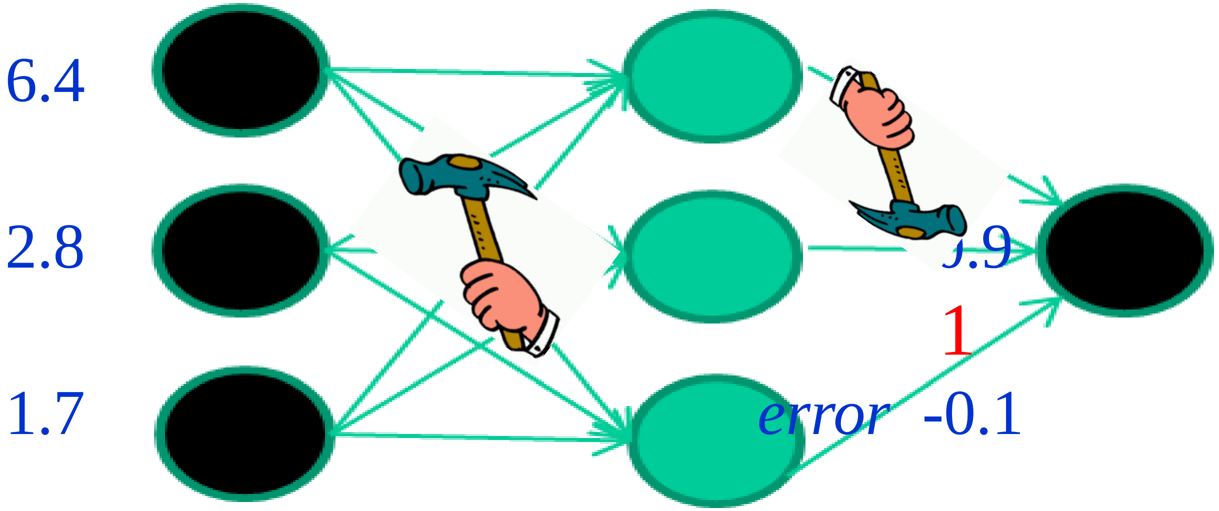
How do they learn?

Source: Prof Corne, Heriot-Watt Unviersity, UK

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

And so on ….

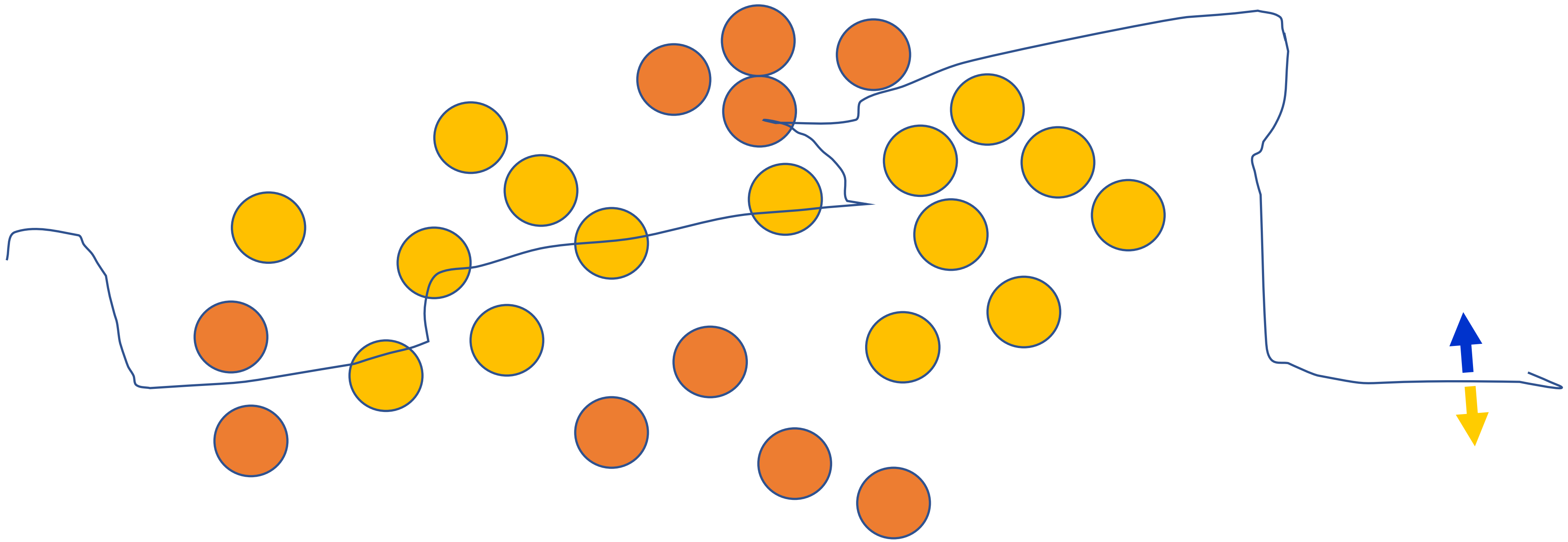


Repeat this thousands, maybe millions of times – each time
taking a random training instance, and making slight
weight adjustments, reduce the error

Called “Gradient Descent”

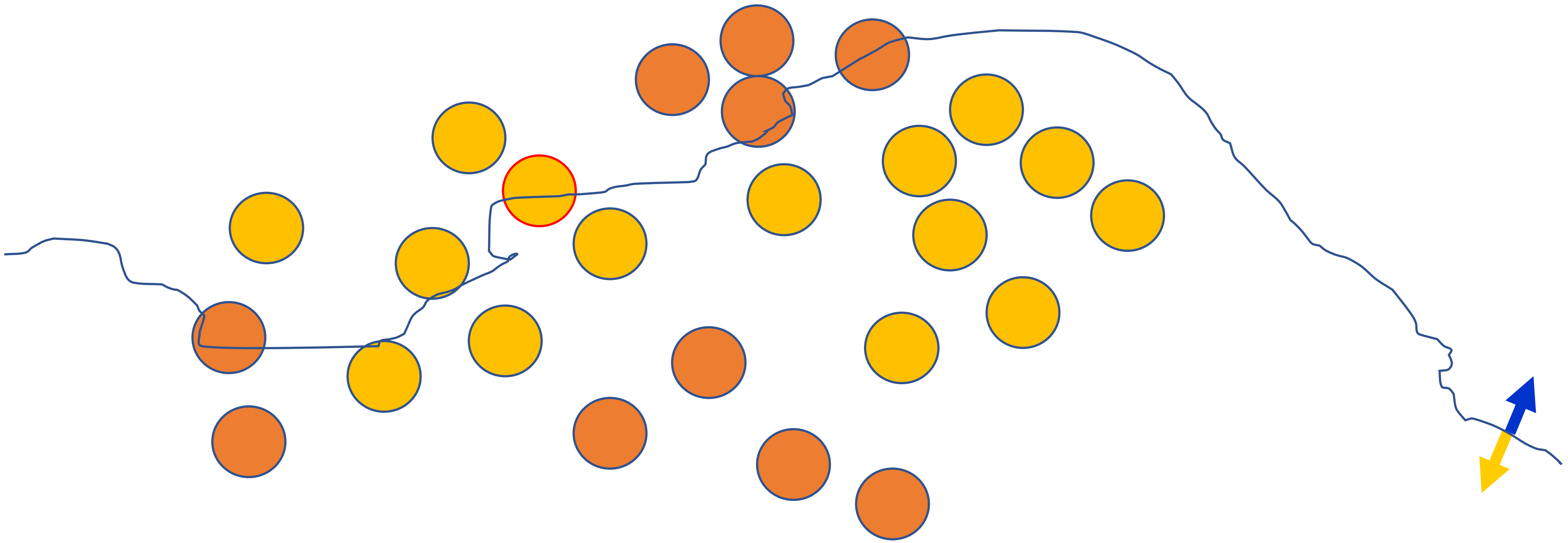
The decision boundary perspective…

Initial random weights



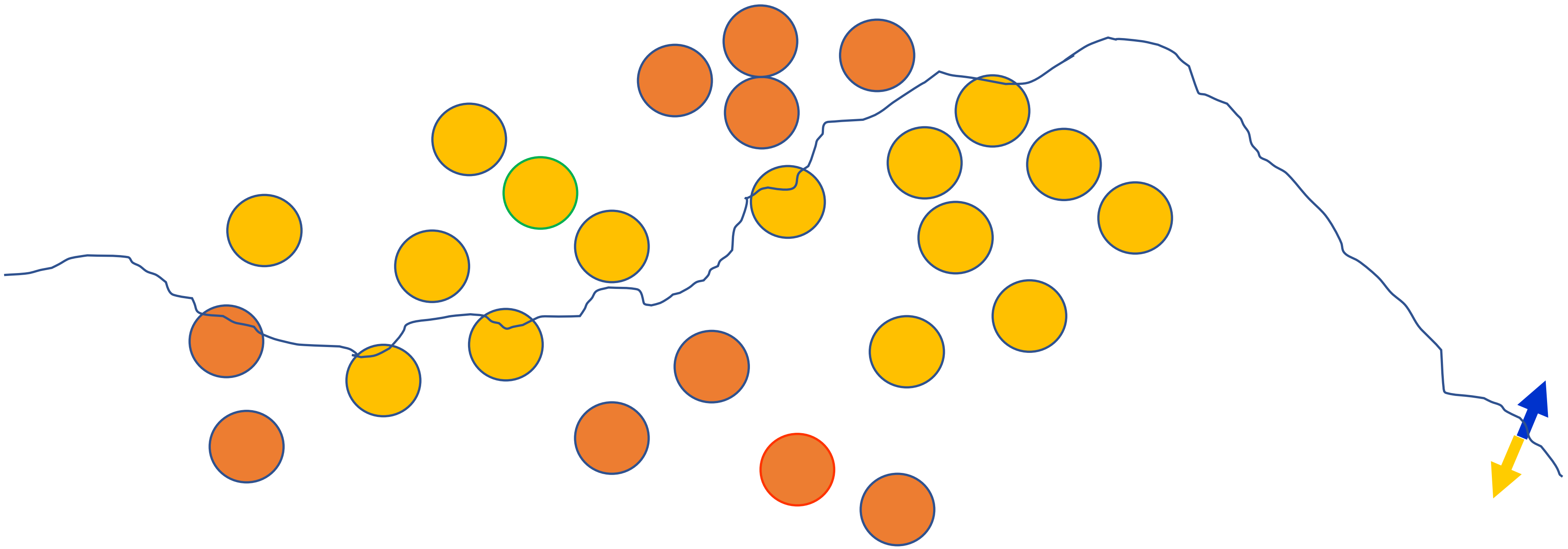
The decision boundary perspective…

Present a training instance / adjust the weights



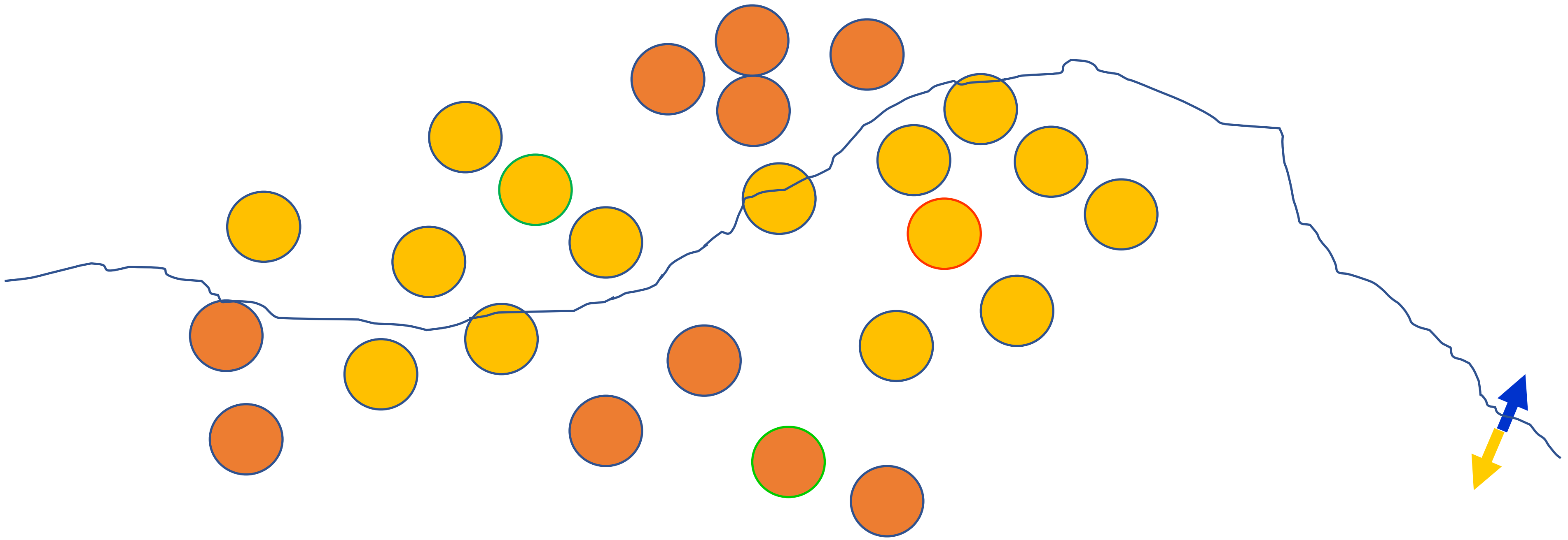
The decision boundary perspective…

Present a training instance / adjust the weights



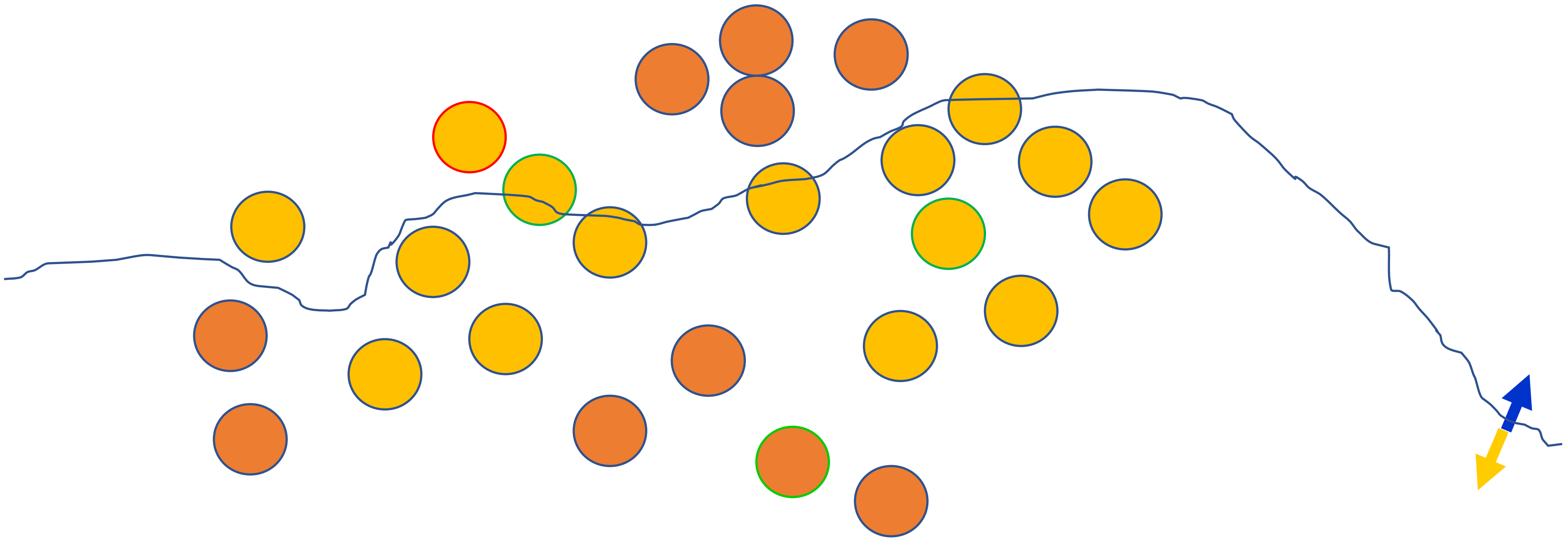
The decision boundary perspective…

Present a training instance / adjust the weights



The decision boundary perspective…

Present a training instance / adjust the weights



The decision boundary perspective…

Eventually ….