CDAC DEEP LEARNING WORKSHOP

Introduction to Deep Learning

24 Aug 2016

Vineeth N Balasubramanian
Indian Institute of Technology, Hyderabad

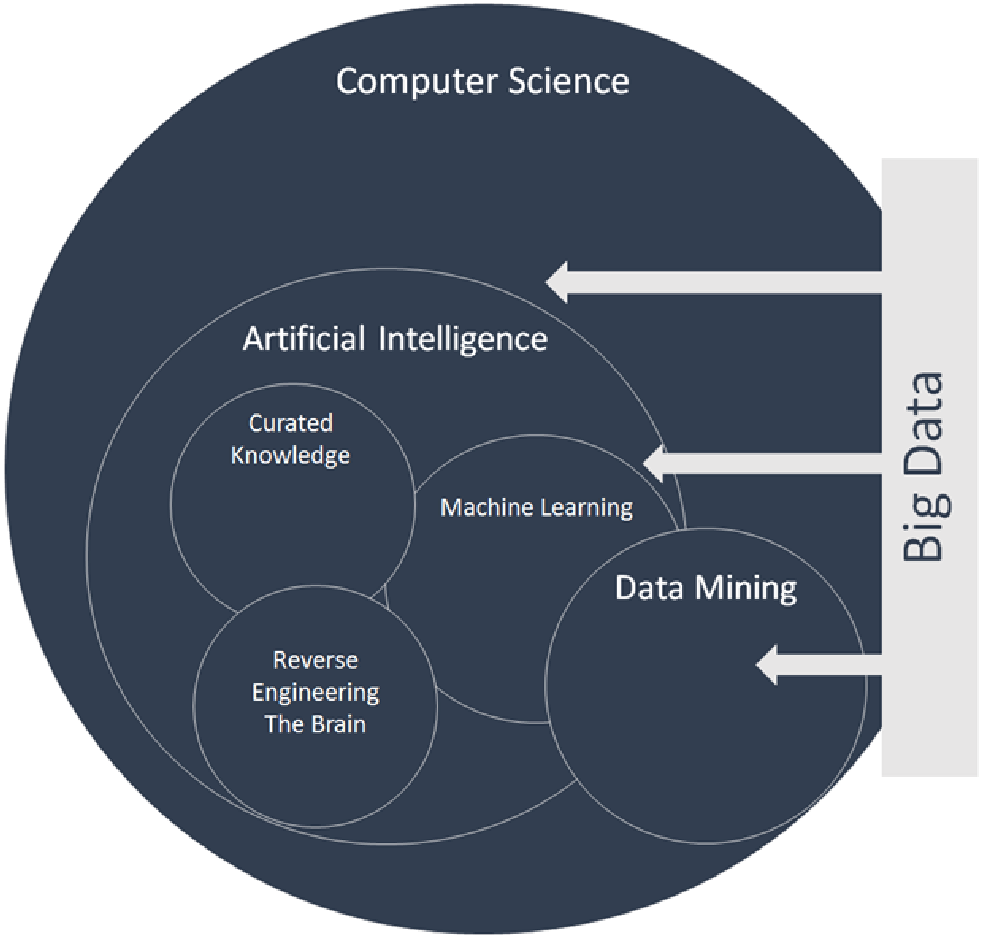
आई आई टी हैदराबाद
IIT Hyderabad

Relevance

AlphaGo: The Recent Sensation

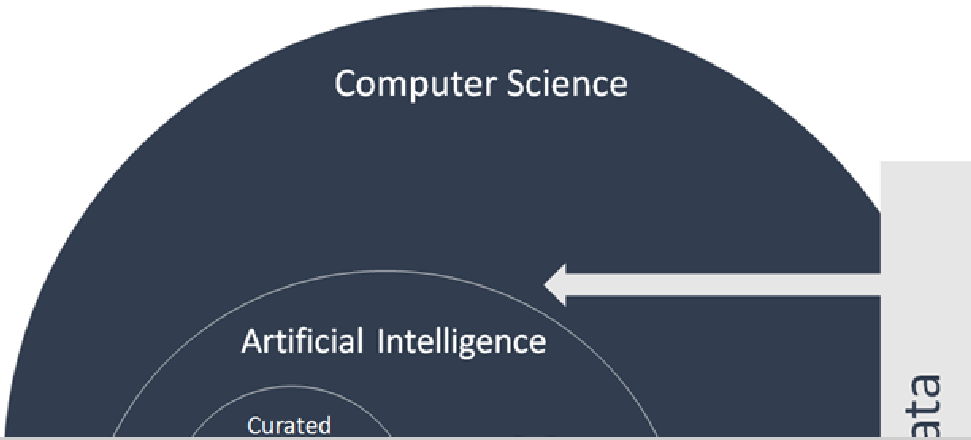
Machine Learning (vs) Artificial Intelligence

Introduction

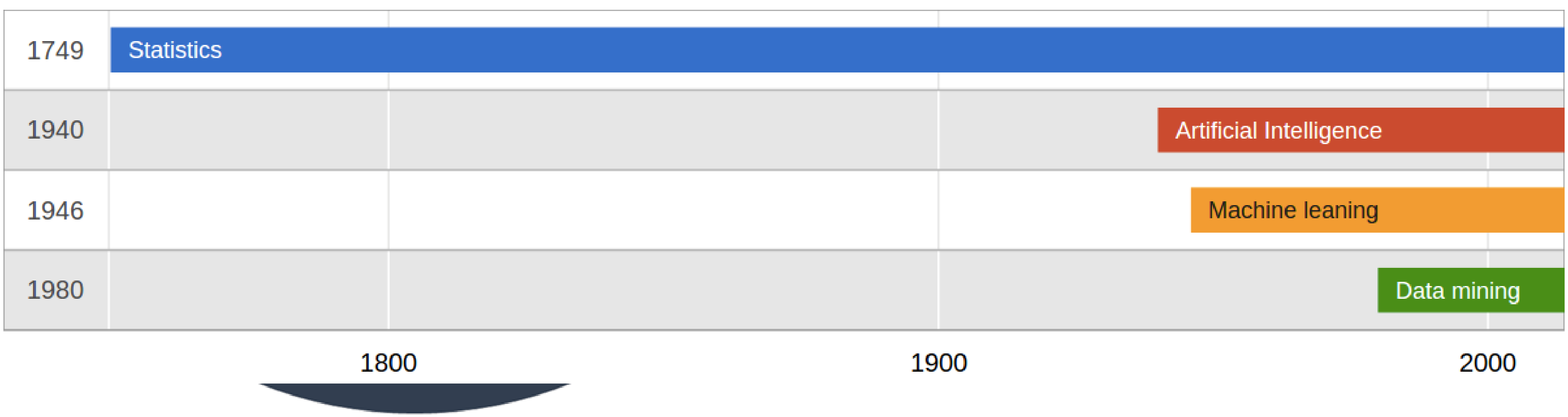


Machine Learning (vs) Artificial Intelligence

Introduction



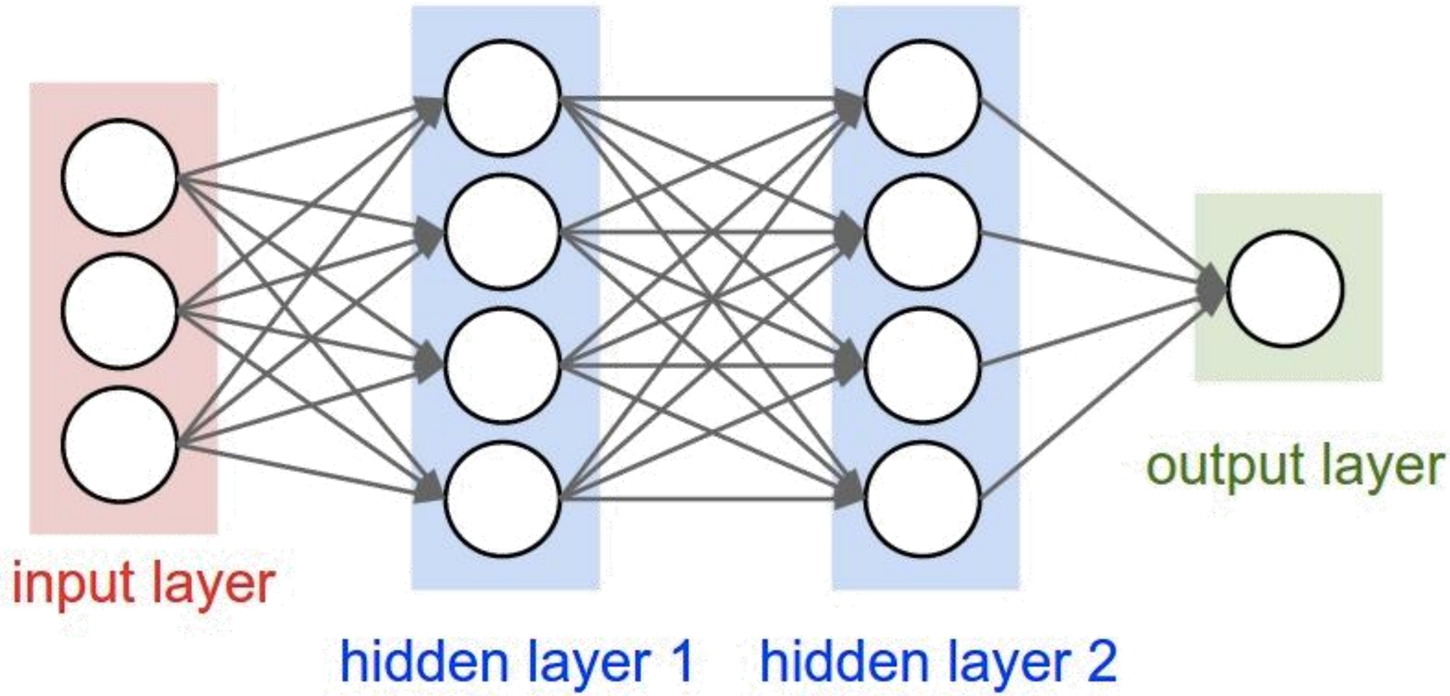
Deep learning: A sub-area of machine learning, that is today understood as representation learning



Deep Learning

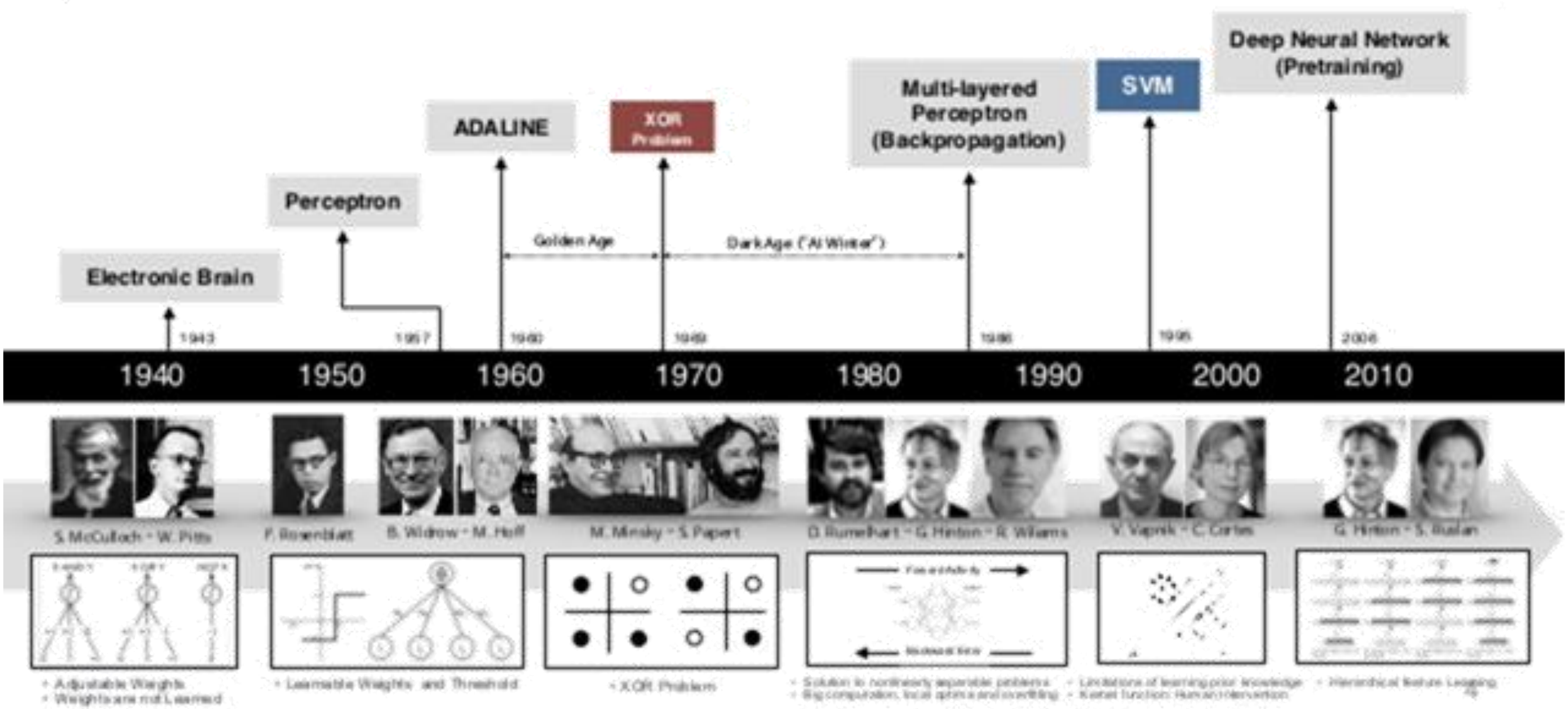
Introduction

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



Deep Learning

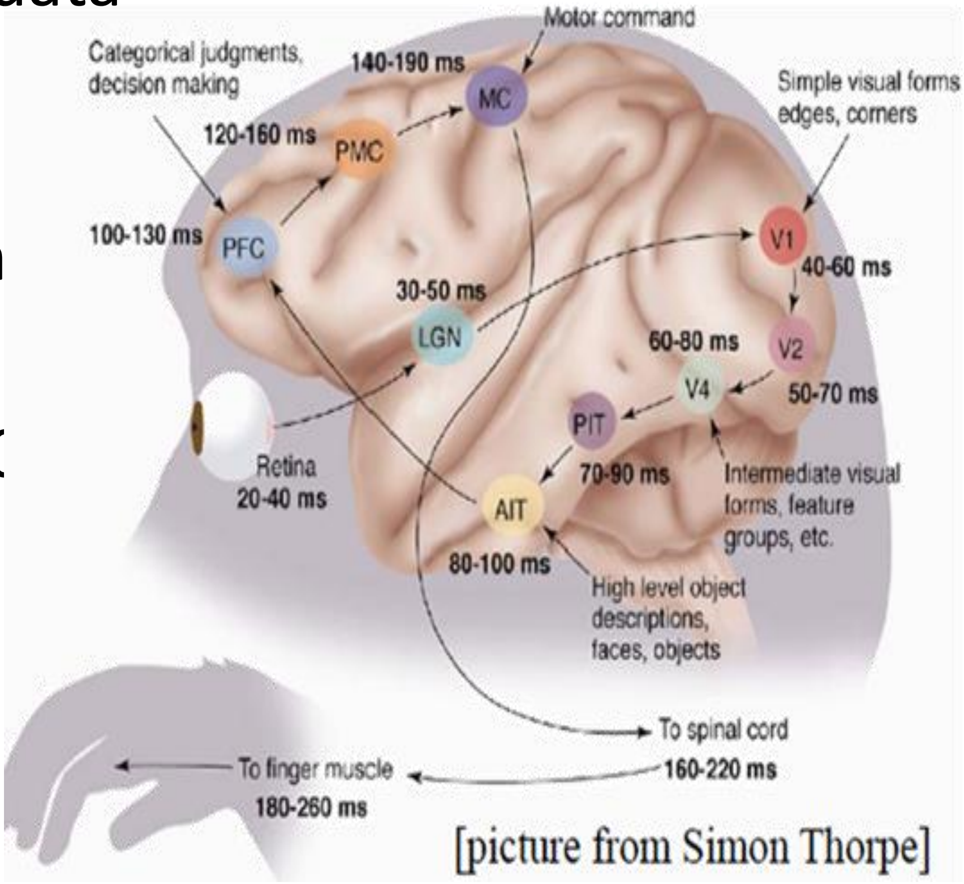
History



Deep Learning

Why is it successful?

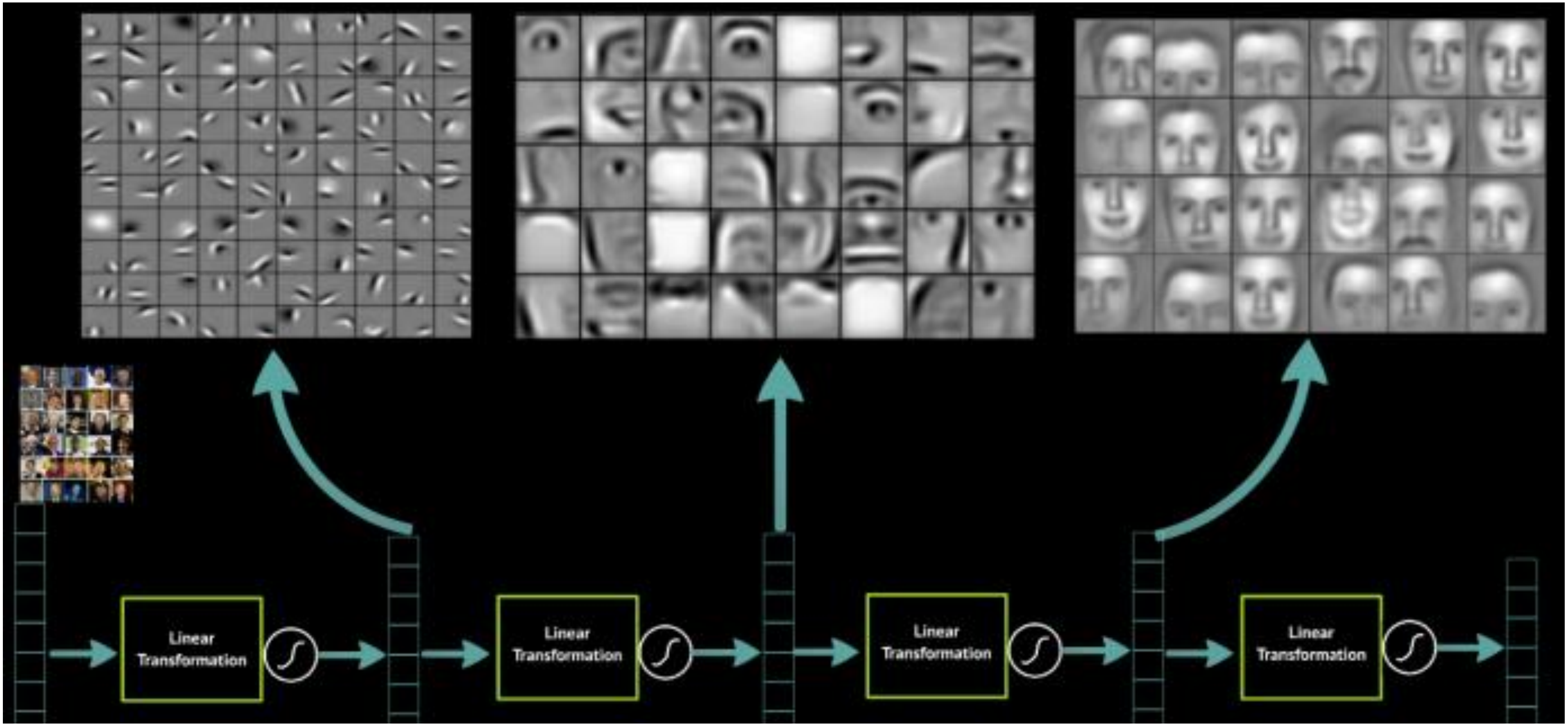
- Learns representations of data that are useful (Other ML algorithms are “shallow”)
- Similar to the human brain
- Then, why was it not successful earlier (in the 9(
  - Computational power
  - Data power



[picture from Simon Thorpe]

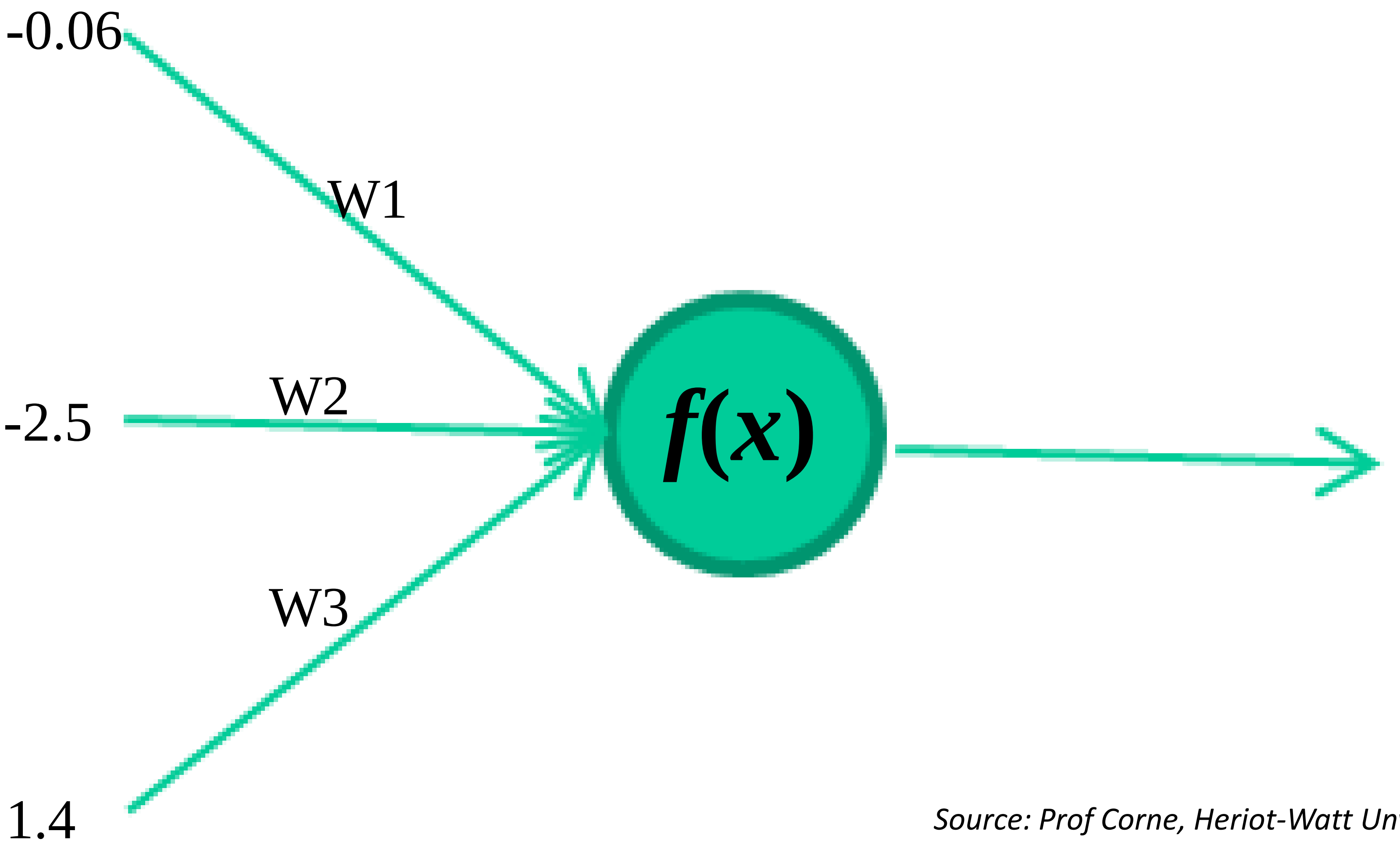
Deep Learning

Why is it successful?



Deep Learning

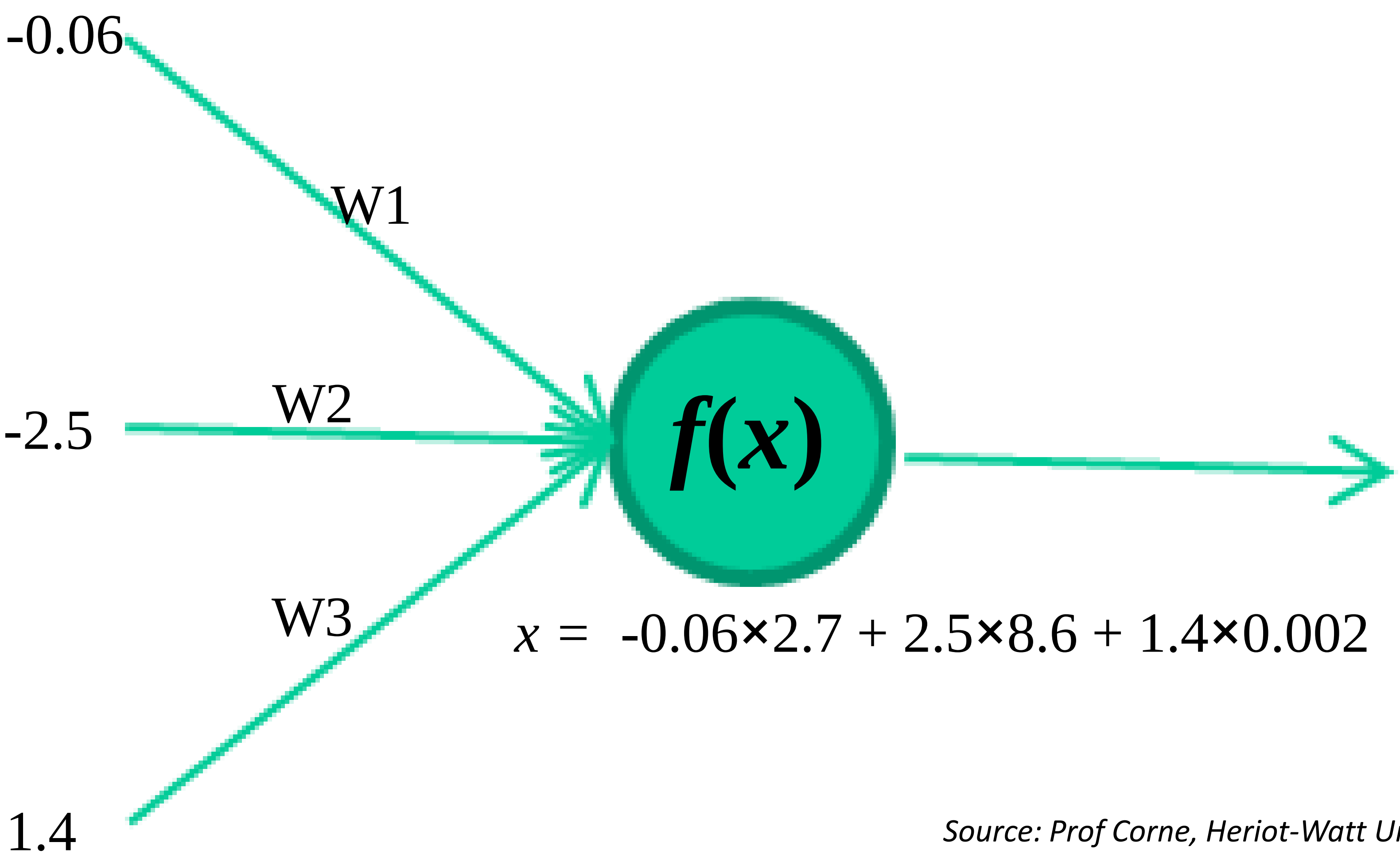
How do they learn?



Source: Prof Corne, Heriot-Watt Unviersity, UK

Deep Learning

How do they learn?



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

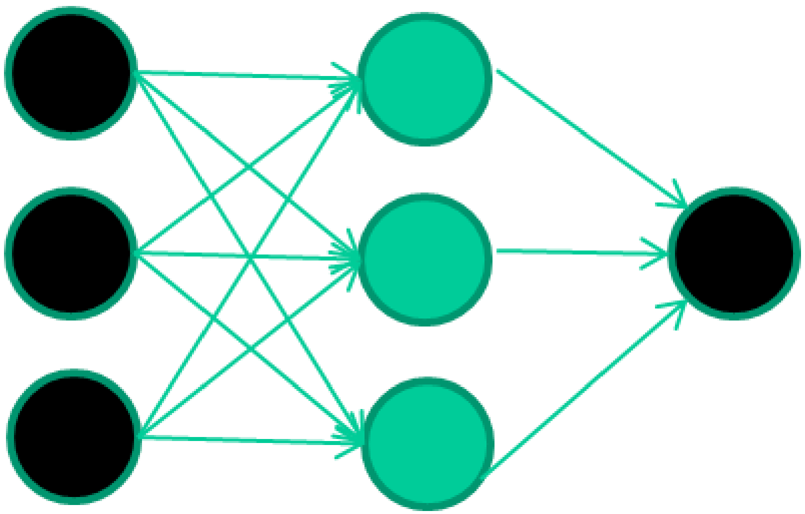
Source: Prof Corne, Heriot-Watt Unviersity, UK

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

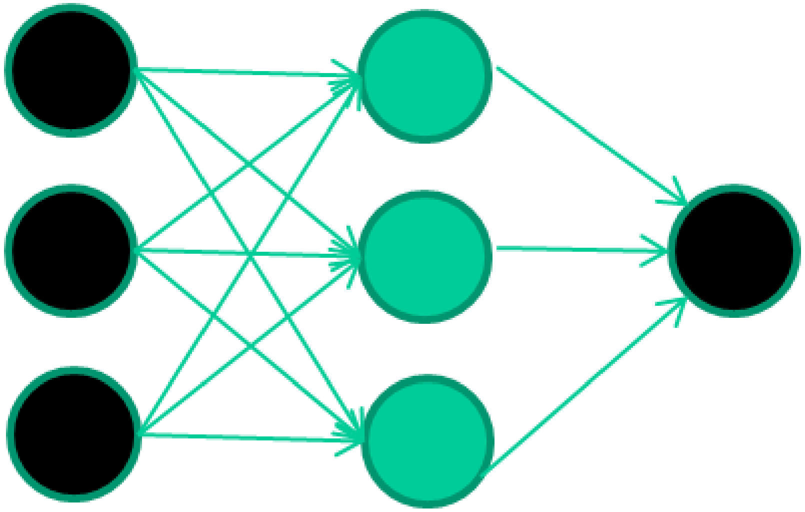
Deep Learning

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



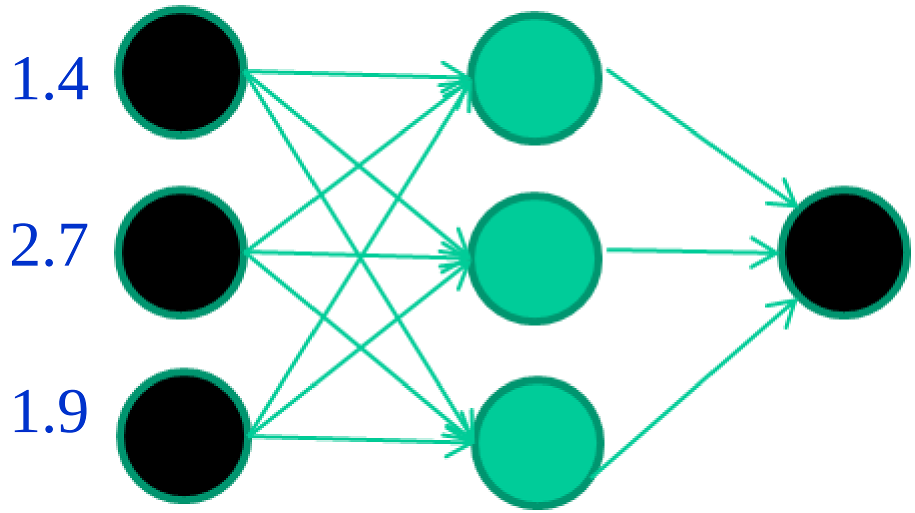
Source: Prof Corne, Heriot-Watt Unviersity, UK

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

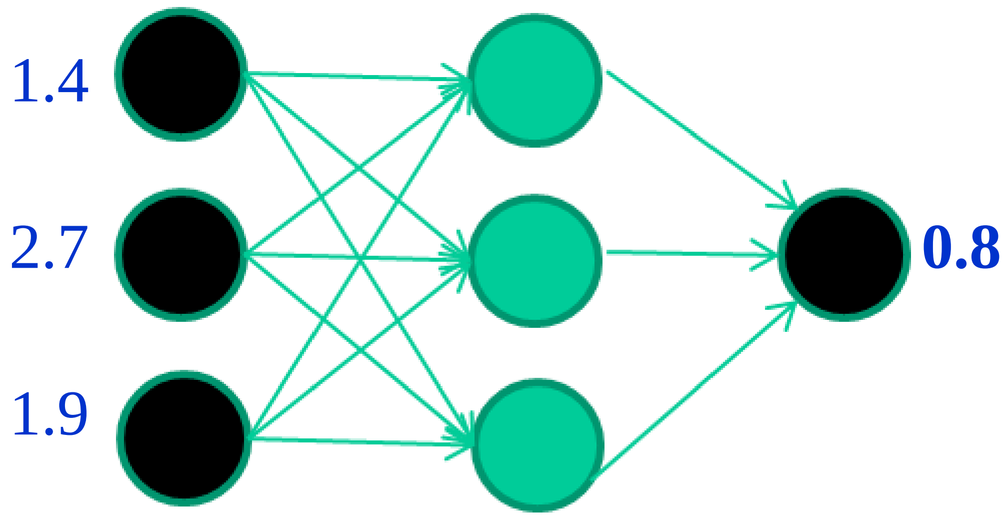
Deep Learning

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

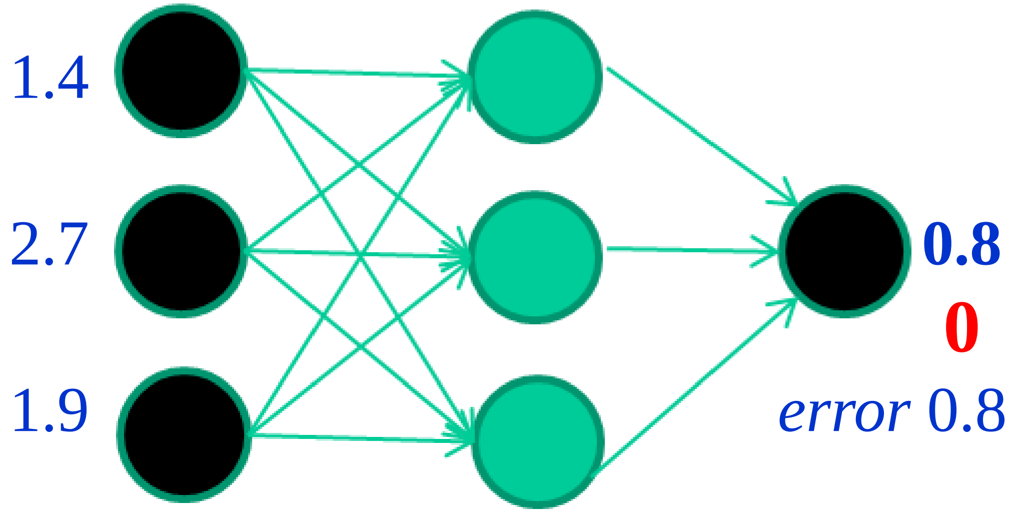
How do they learn?

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

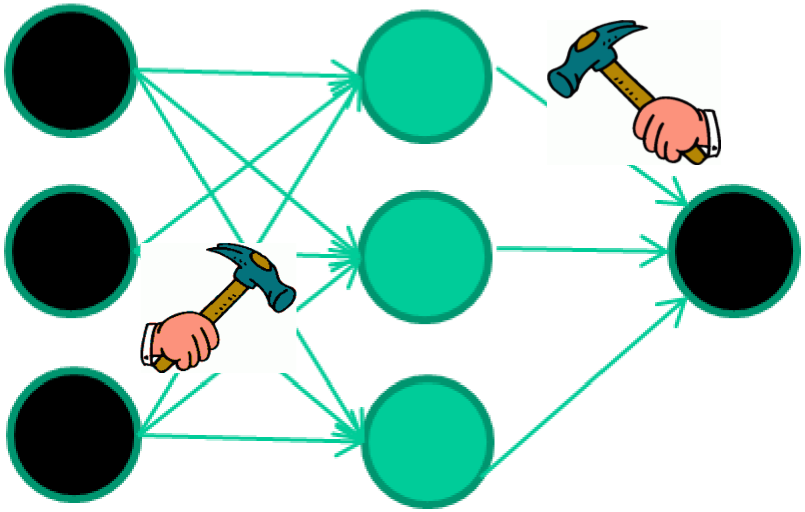
Deep Learning

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

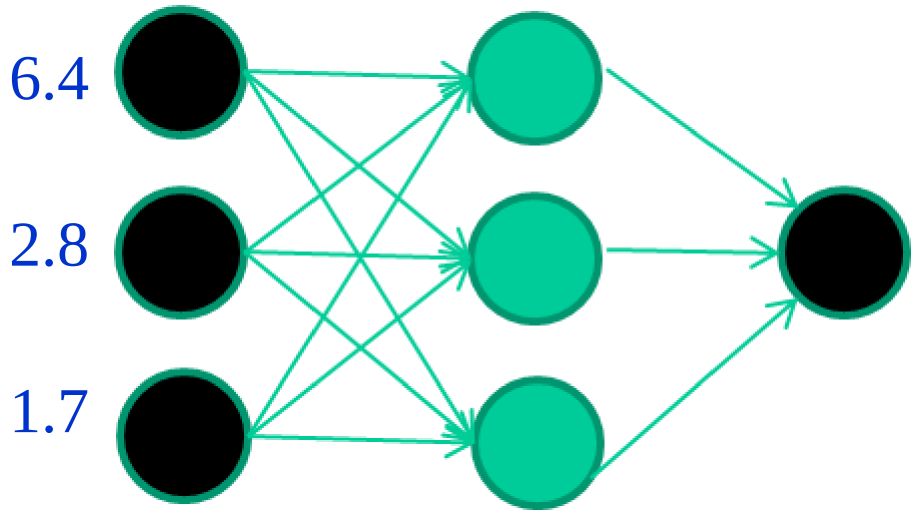
Deep Learning

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

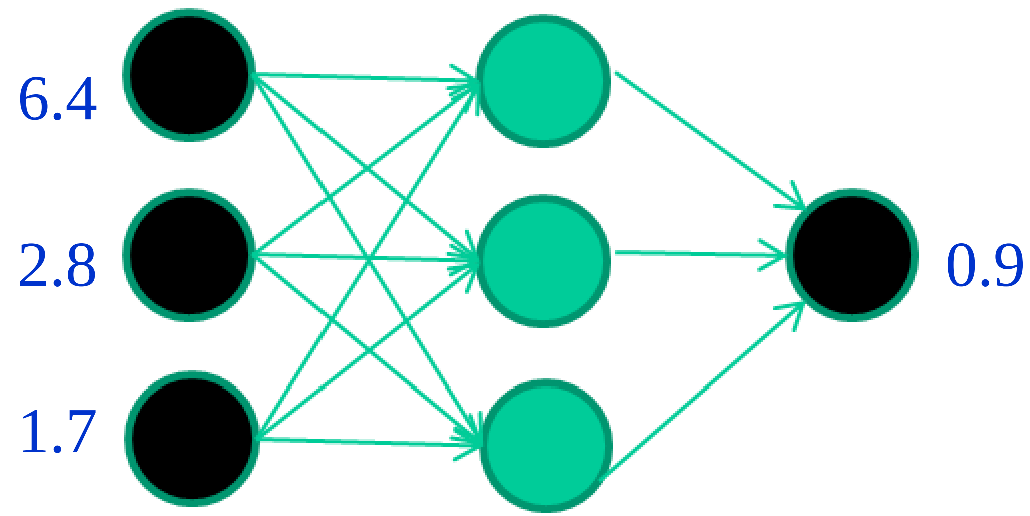
How do they learn?

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

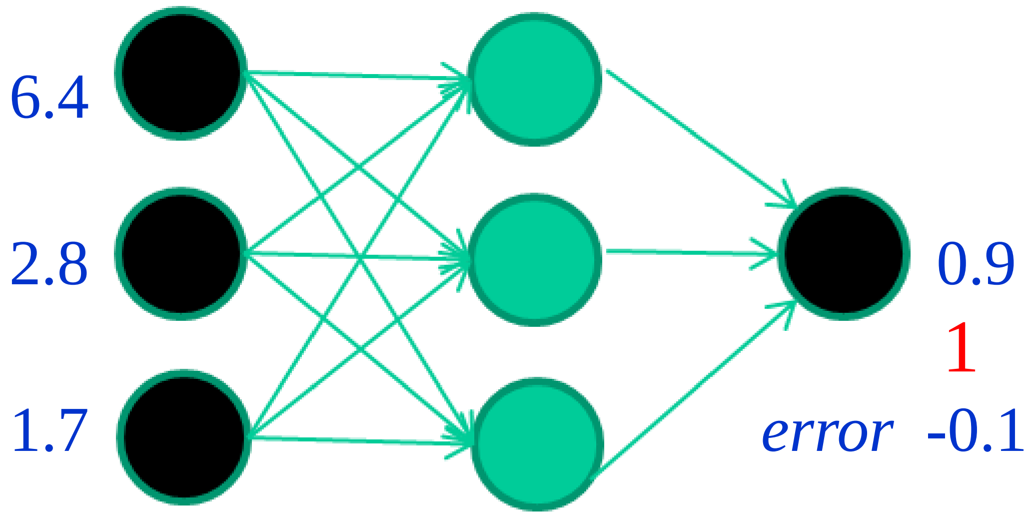
Deep Learning

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

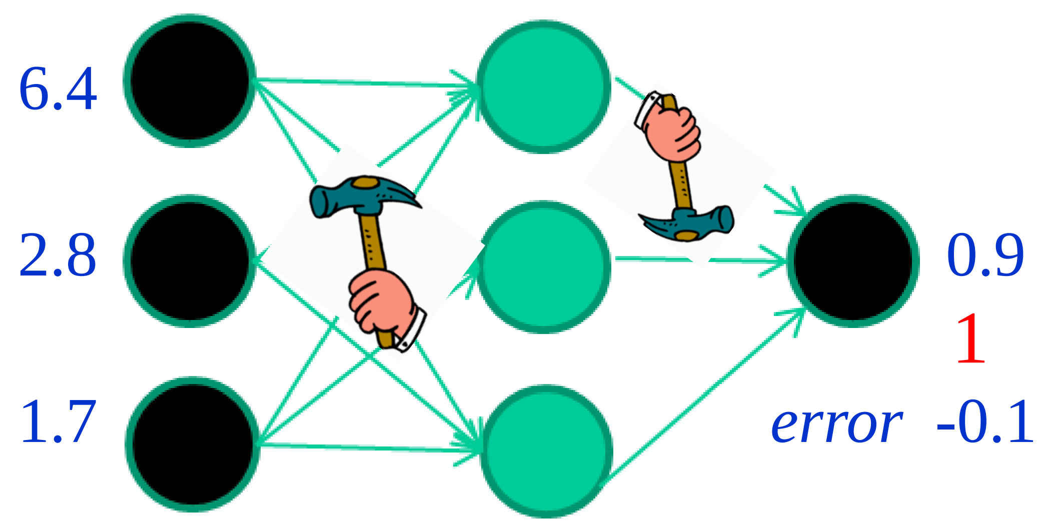
Deep Learning

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

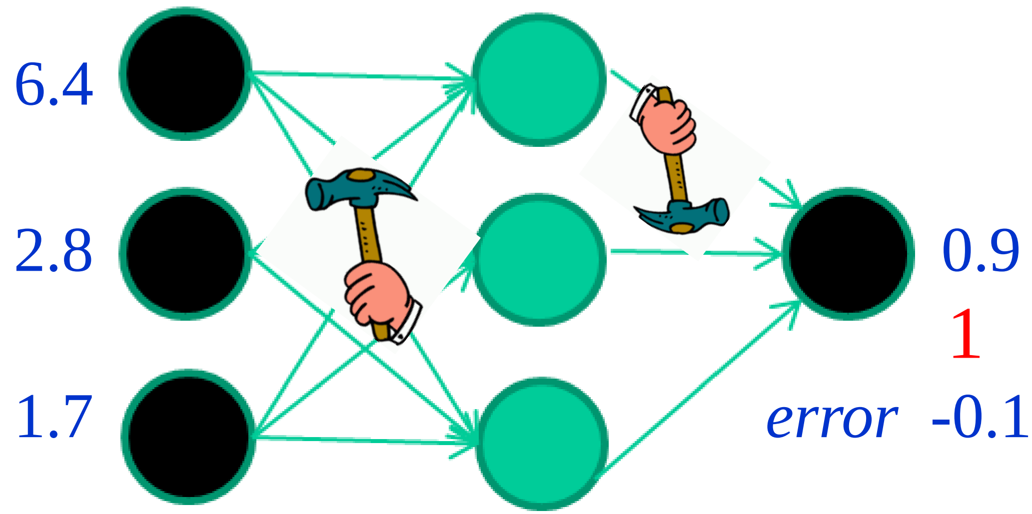
How do they learn?

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 ….

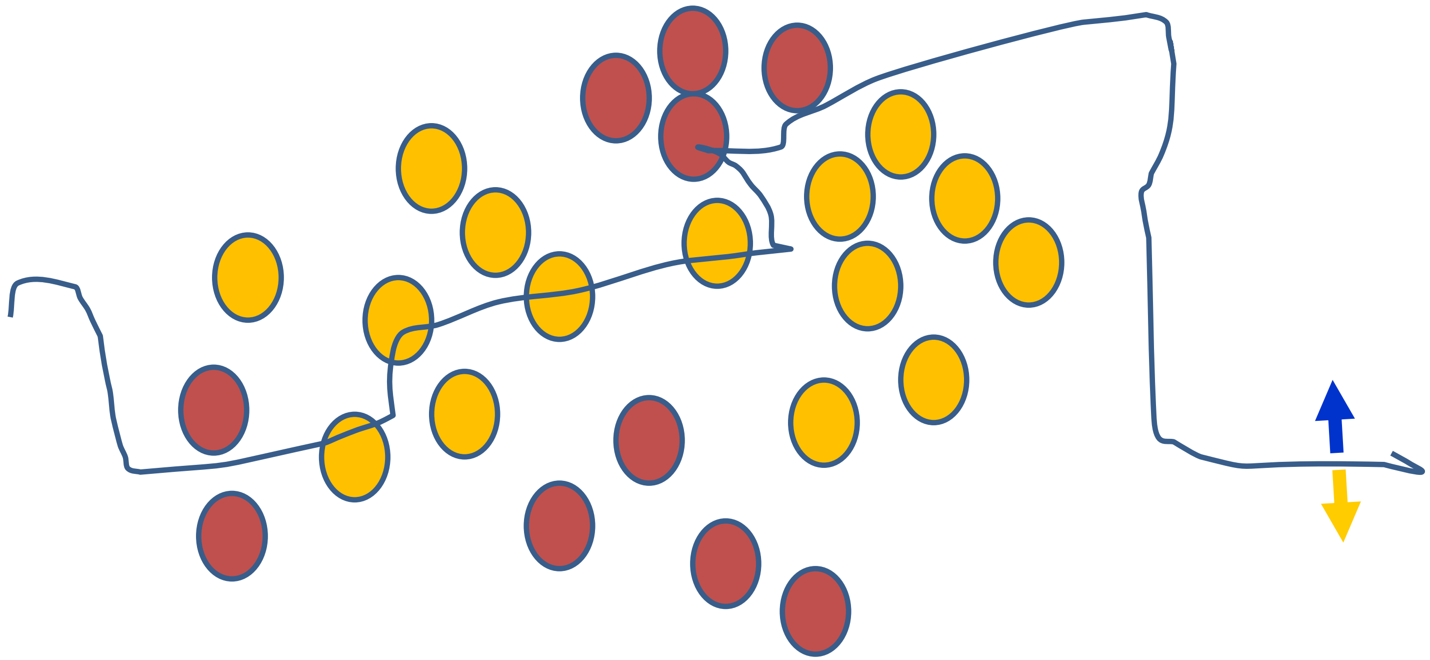


Repeat this thousands, maybe millions of times – each time taking a random training instance, and making slight weight adjustments, reduce the error

Called "Gradient Descent"

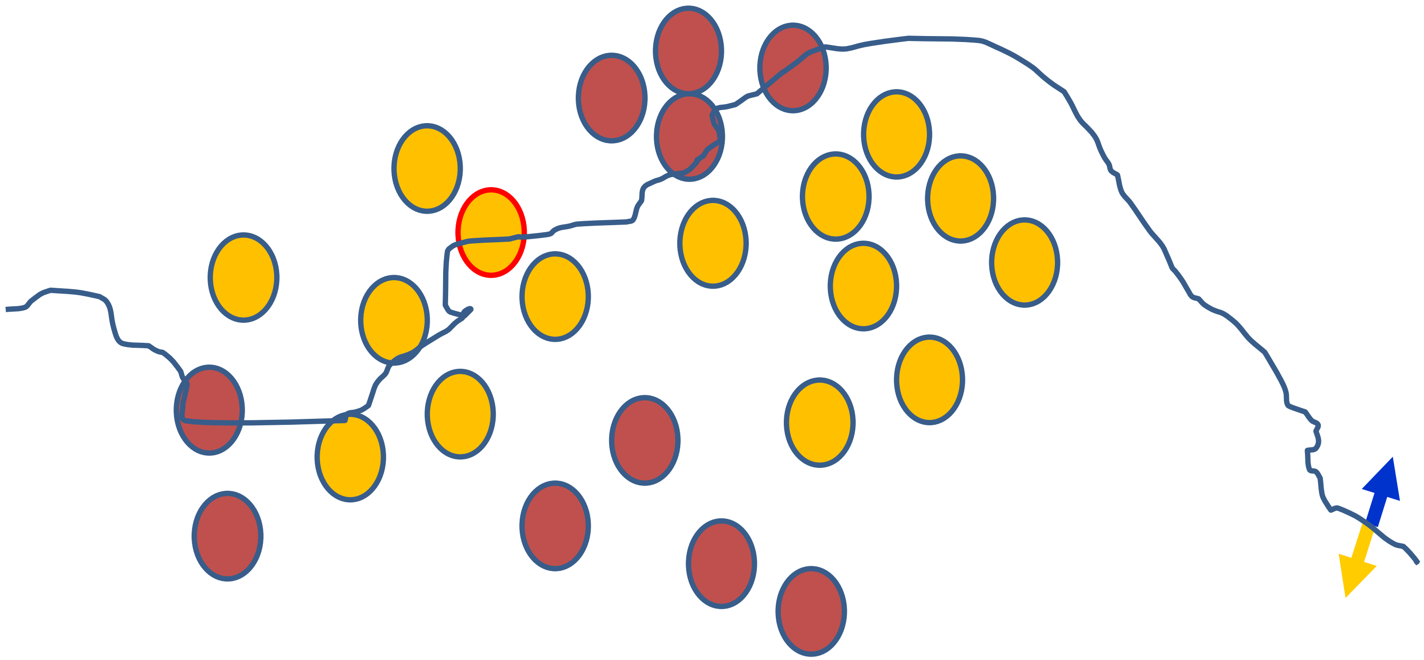
The decision boundary perspective…

Initial random weights



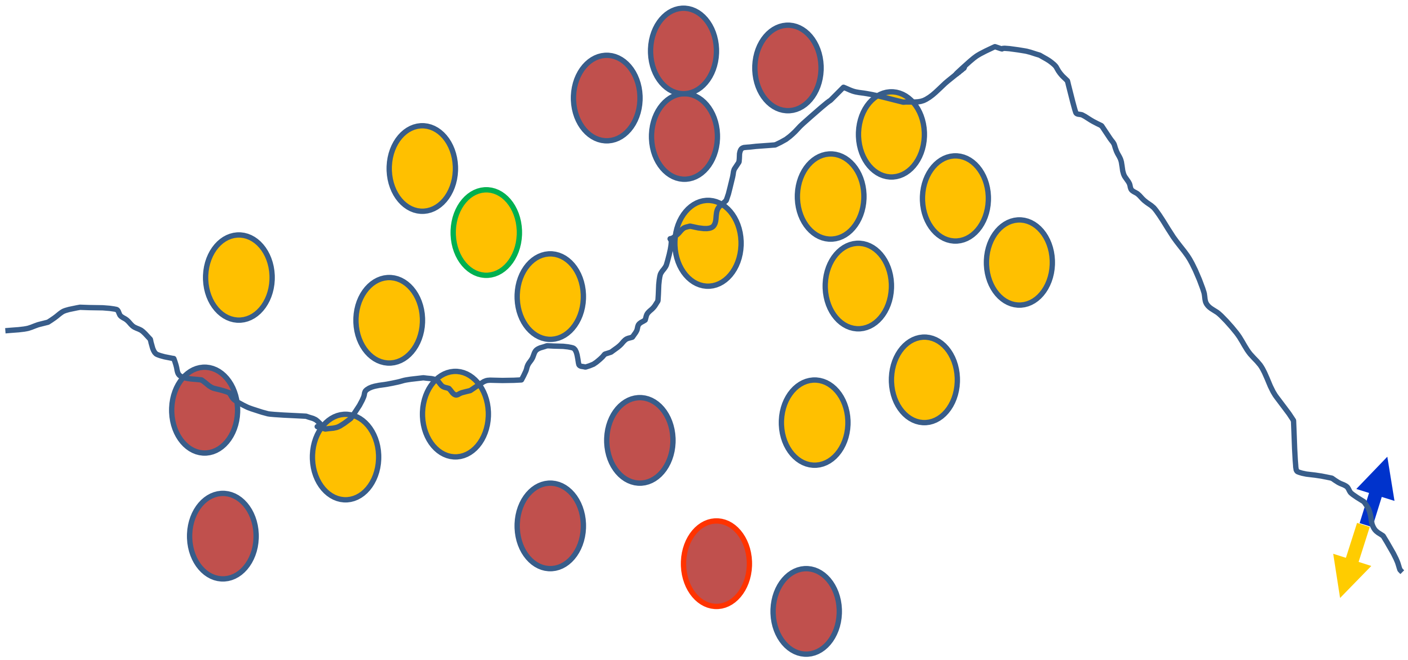
The decision boundary perspective…

Present a training instance / adjust the weights



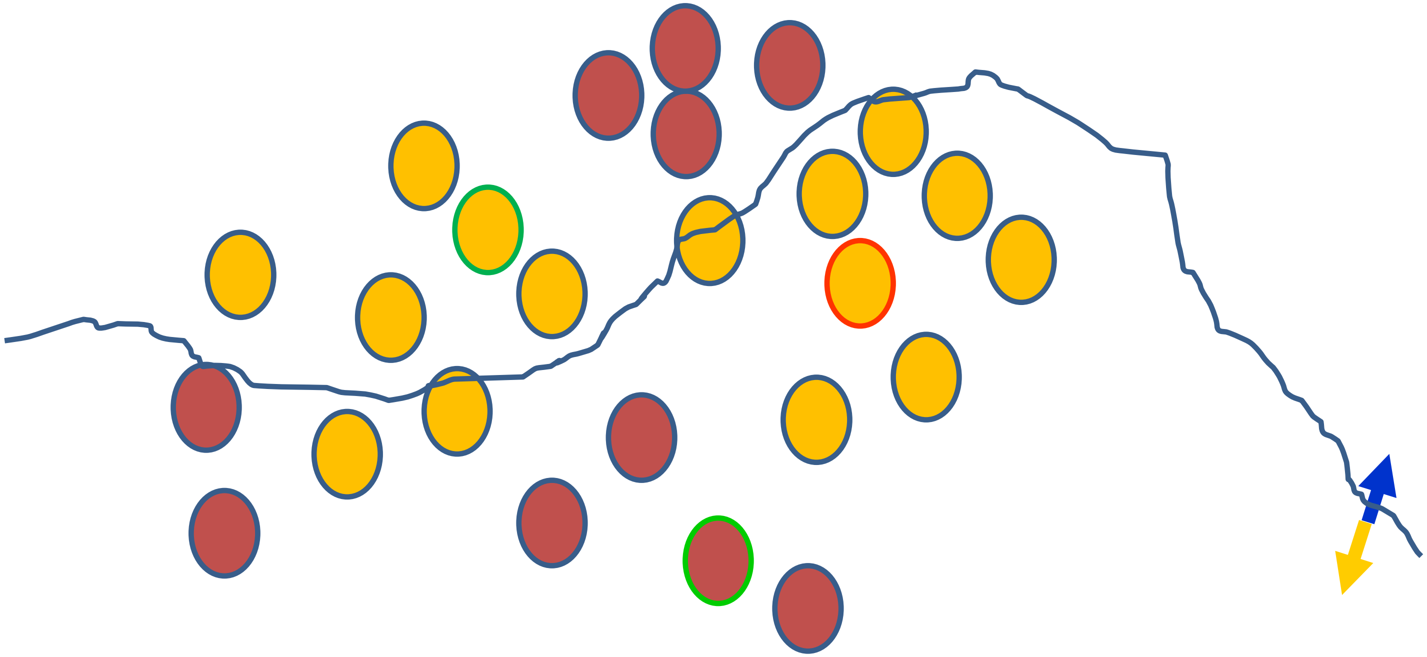
The decision boundary perspective…

Present a training instance / adjust the weights



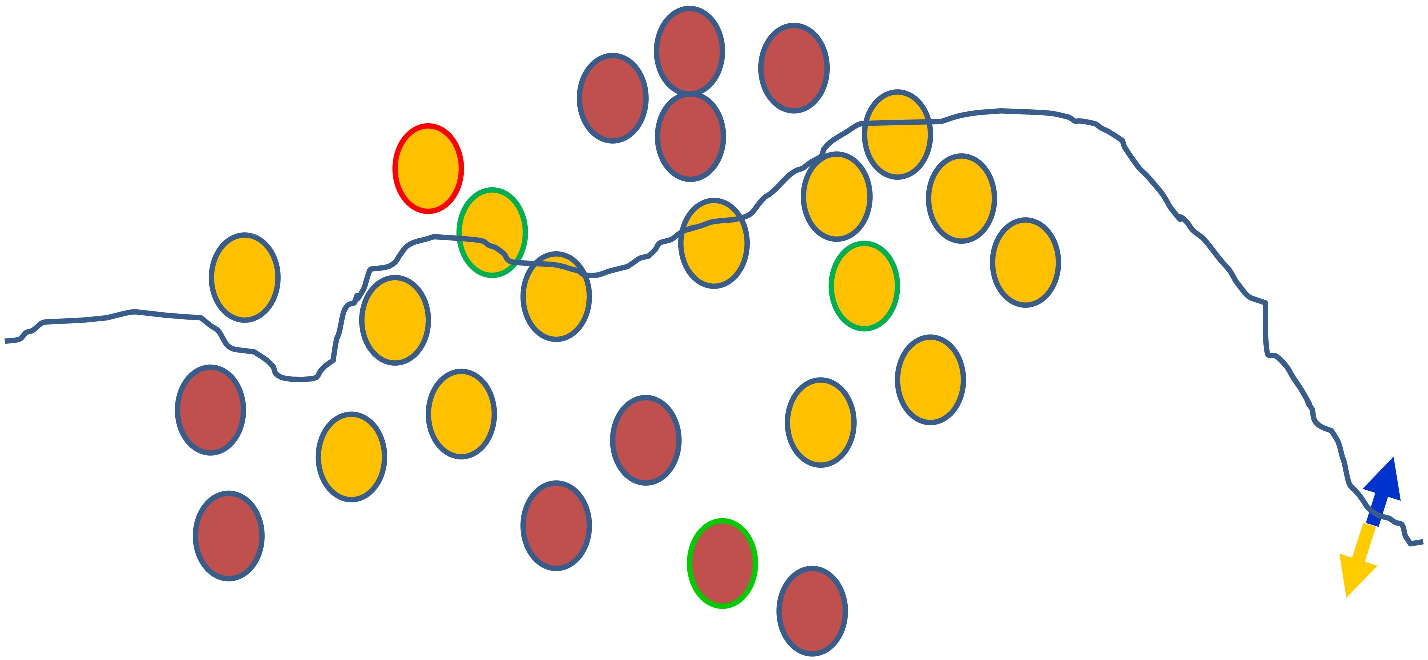
The decision boundary perspective…

Present a training instance / adjust the weights



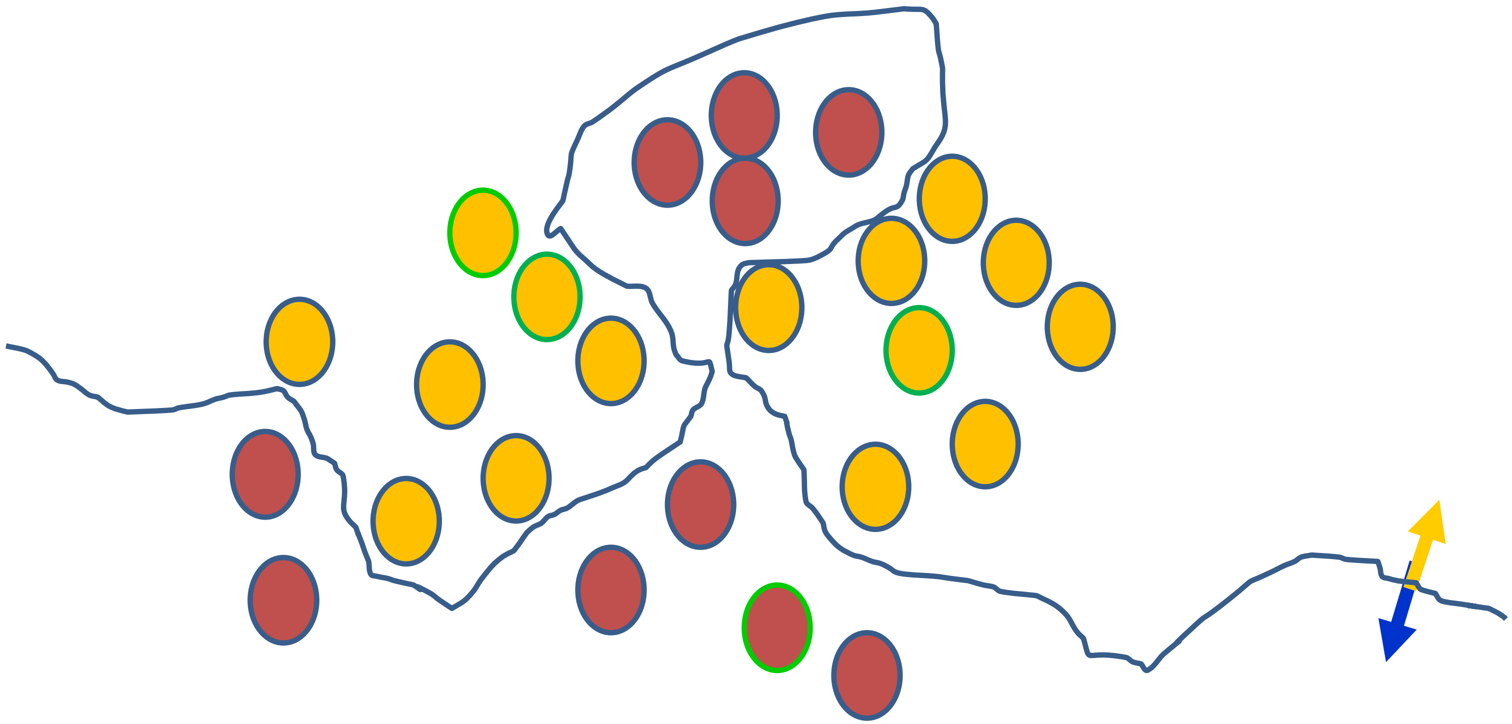
The decision boundary perspective…

Present a training instance / adjust the weights

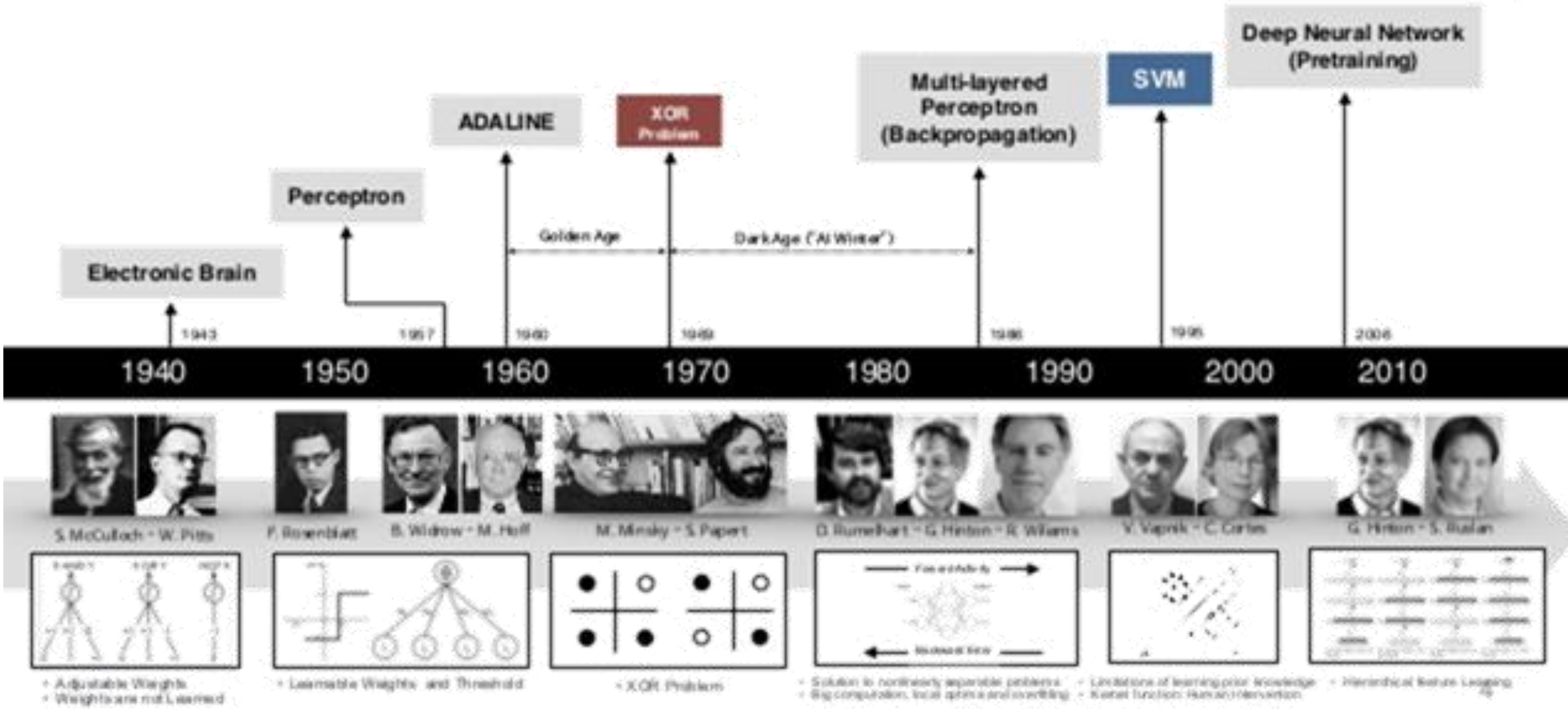


The decision boundary perspective…

Eventually ….



History of Neural Networks



Neural Networks Training: Backpropagation

- Extension of perceptrons to multiple layers
- 1. Initialize network with random weights
- 2. For all training cases (called examples):
  - **a.** Present training inputs to network and calculate output
  - **b.** For all layers (starting with output layer, back to input layer):
    - i. Compare network output with correct output (error function)
    - ii. Adapt weights in current layer

Neural Networks Training: Backpropagation

- Method for learning weights in feed-forward (FF) nets

- Can't use Perceptron Learning Rule
  - no teacher values are possible for hidden units

- Use gradient descent to minimize the error
  - propagate deltas to adjust for errors
    backward from outputs
      to hidden layers
        to inputs

Neural Networks Training: Backpropagation

- The idea of the algorithm can be summarized as follows :

1. Computes the **error term for the output units** using the observed error.

- 2. From output layer, repeat
  - propagating the error term back to the previous layer and updating the weights between the two layers until the earliest hidden layer is reached.

Neural Networks Training: Backpropagation

- Initialize weights (typically random!)
- Keep doing epochs
  - For each example **e** in training set do
    - **forward pass** to compute
      - O = neural-net-output(network,e)
      - miss = (T-O) at each output unit
    - **backward pass** to calculate deltas to weights
    - update all weights
  - end
- until tuning set error stops improving

Forward pass

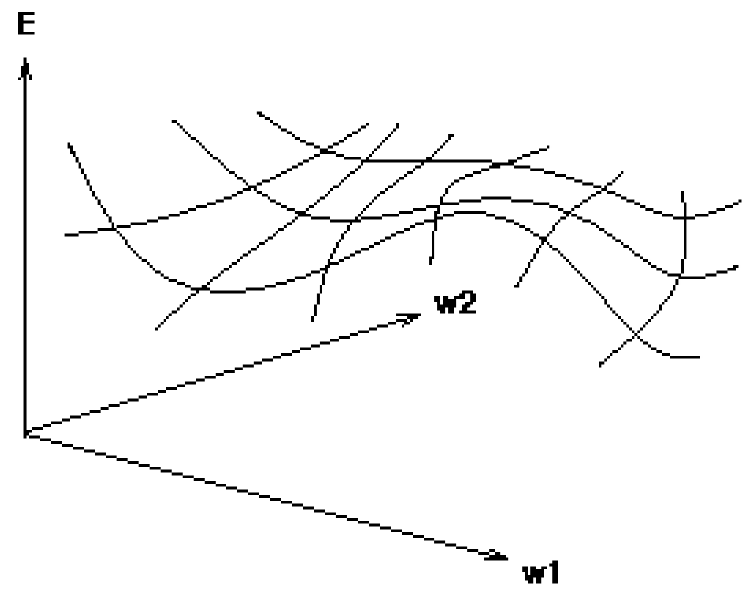
Backward pass explained in next slide

आई आई टी हैदराबाद
IIT Hyderabad

Neural Networks Training: Backpropagation

Gradient Descent

- Think of the N weights as a point in an N-dimensional space
- Add a dimension for the observed error
- Try to minimize your position on the "error surface"



Neural Networks Training: Backpropagation

Compute deltas

Gradient Descent

- Trying to make error decrease the fastest
- Compute:
  - $Grad_E$ = [dE/dw1, dE/dw2, . . ., dE/dwn]
- Change i-th weight by
  - $delta_{wi} = -lr * dE/dwi$
  - lr or α learning rate: generally starts with 1E-1 or 10^{-1}

Derivatives of error for weights

- We need a derivative!
- Activation function must be **continuous**, differentiable, non-decreasing, and easy to compute

Neural Networks Training: Backpropagation

Updating Hidden-to-Output

- We have teacher supplied desired values

- $\text{delta}_{wji} = \alpha * a_j * (T_i - Oi) * g'(in_i)$

  $= \alpha * a_j * (T_i - O_i) * O_i * (1 - O_i)$

  - for sigmoid the derivative is, $g'(x) = g(x) * (1 - g(x))$

derivative

alpha

Here we have general formula with derivative, next we use for sigmoid

miss

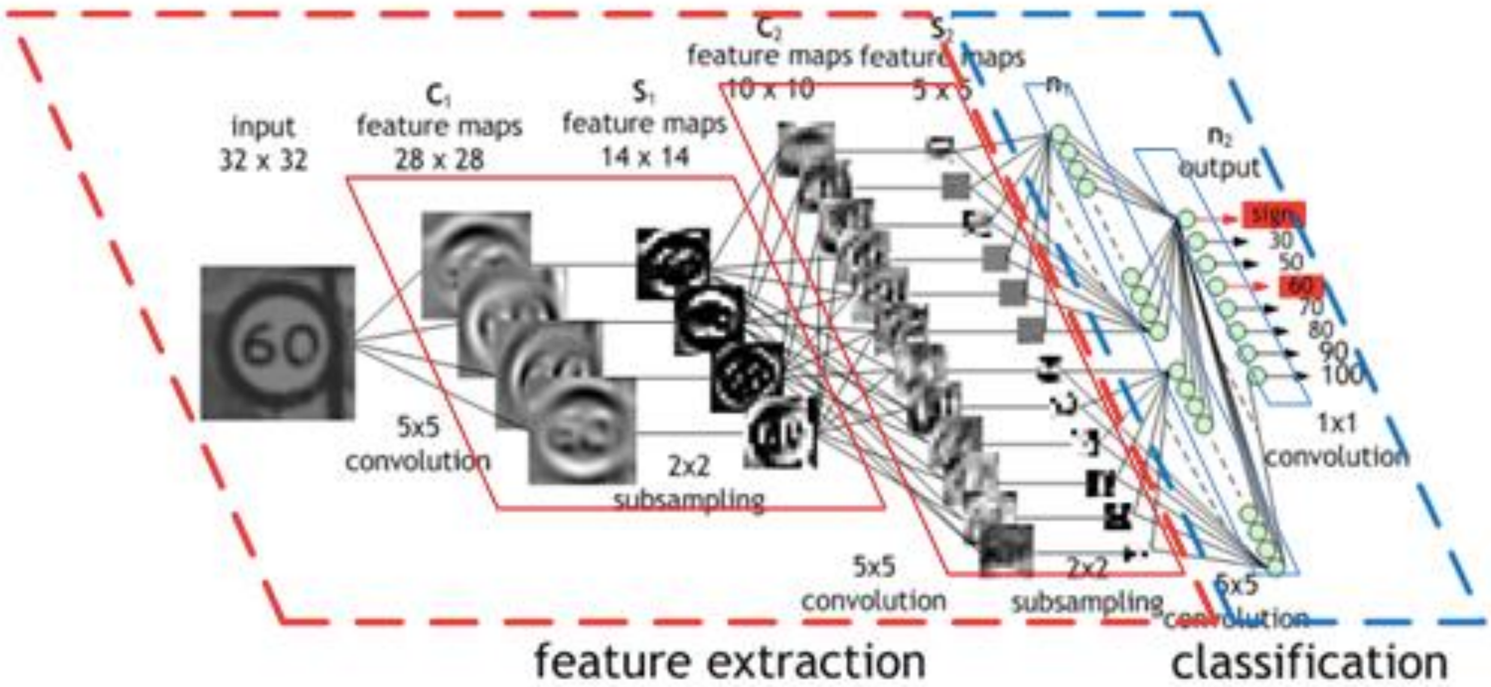
Making Choices

- How do we pick **α or lr**?
  - Tuning set, or
  - Cross validation, or
  - Small for slow, conservative learning
  - for Deep learning typically is 1e-1
- How many hidden layers?
  - Too few ==> can't learn
  - Too many ==> poor generalization
- How big a training set?
  - Determine your target error rate, *e; *Success rate is 1- *e*
  - Typical training set approx. n/*e*, where n is the number of weights in the net
  - Example:
    - *e* = 0.1, n = 80 weights; Training set size 800

Deep Learning Architectures

Variants

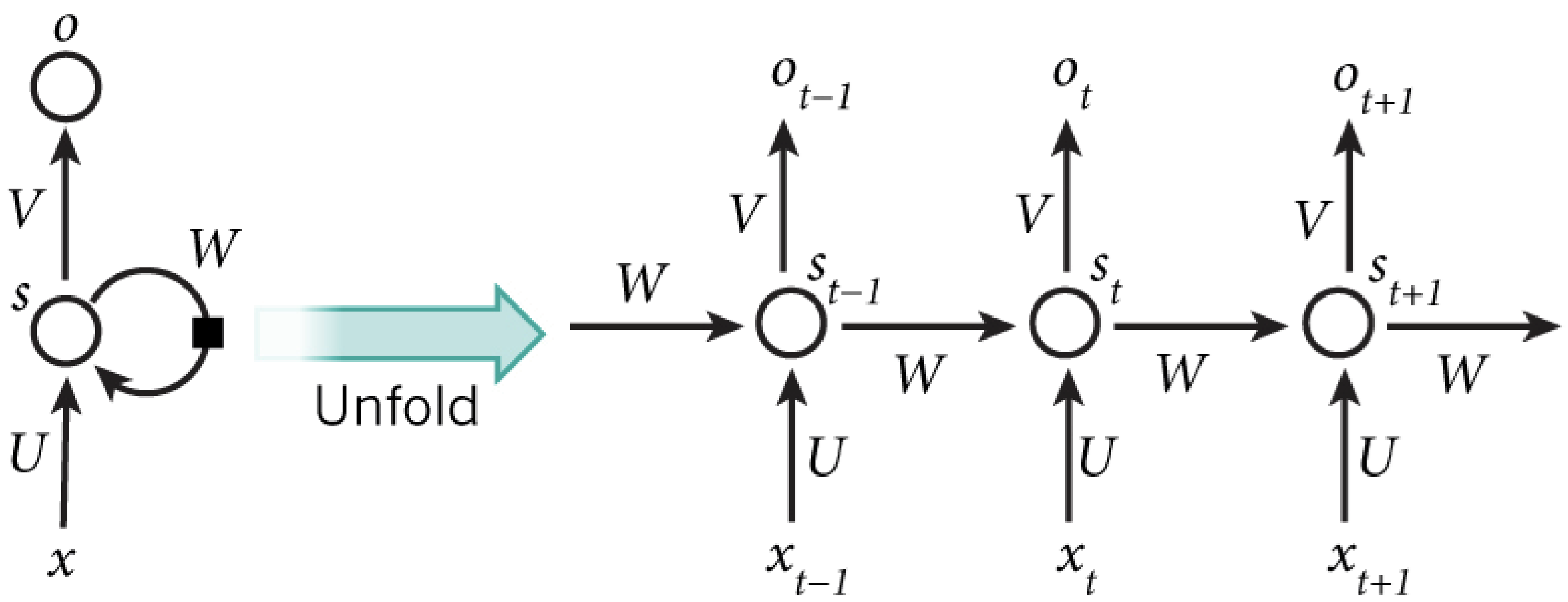
Convolutional Neural Networks for Image and Video Understanding



Deep Learning Architectures

Variants

Recurrent Neural Networks for Time Series and Sequence Data Understanding



Deep Learning

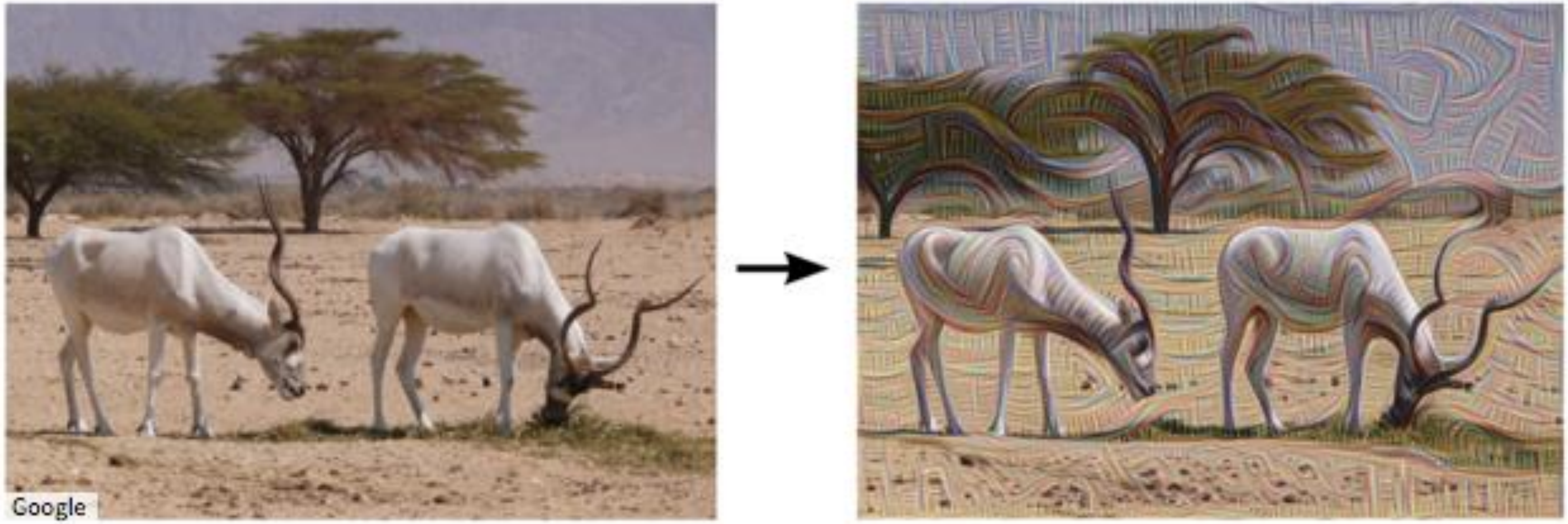
Applications and Successes

- AlexNet (Object Recognition): The network that catapulted the success of deep learning in 2012

Stepping to even higher levels of intelligence

From generation to imagination

- Deep dreaming/hallucination
  - https://www.youtube.com/watch?v=oyxSerkkP4o



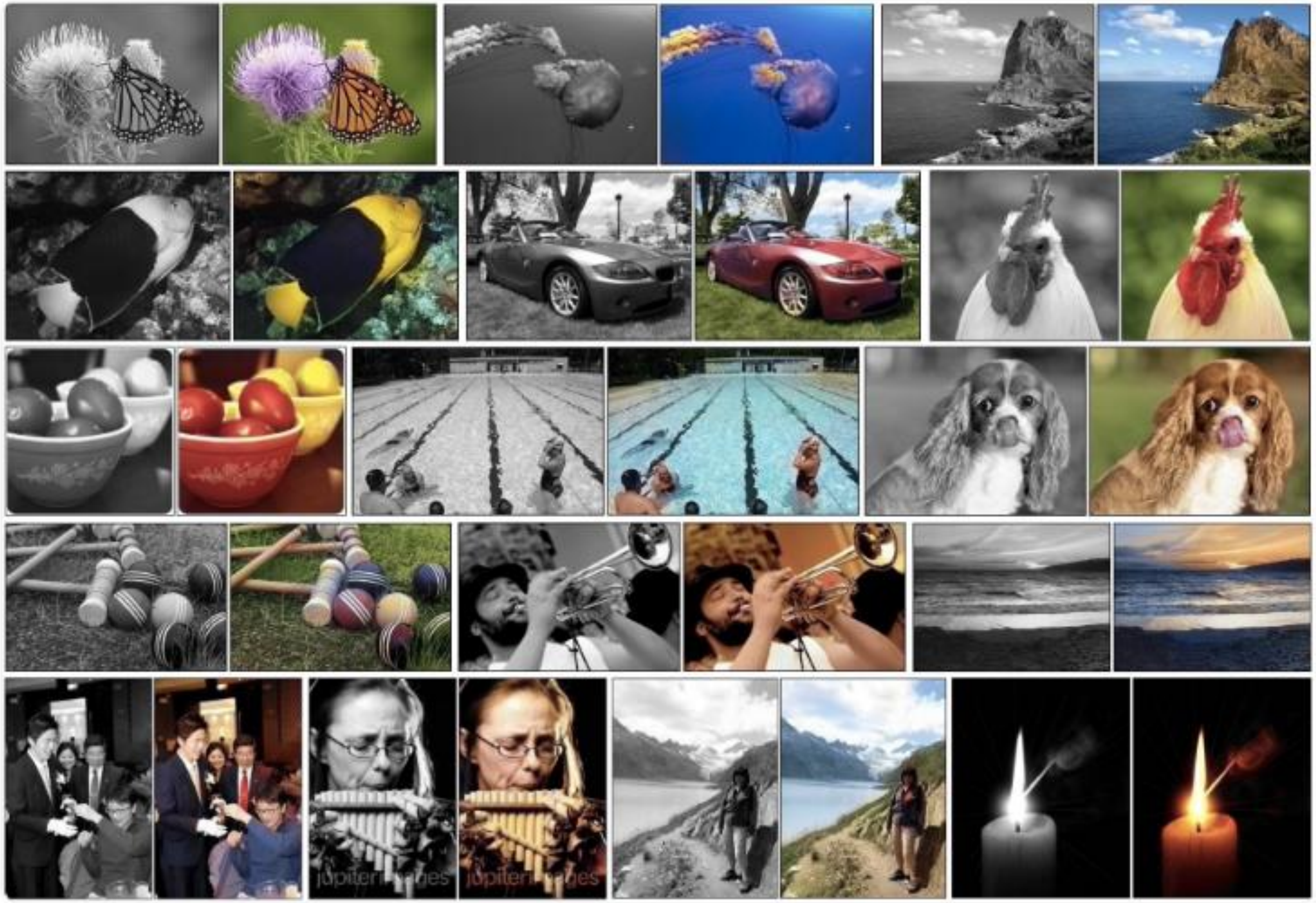
Stepping to even higher levels of intelligence

From generation to imagination

Stepping to even higher levels of intelligence

From generation to imagination

- Deep colorization



What’s cutting edge?

Recent Trends

- Limitless applications
  - Any application where you would like to learn from data
  - When do you choose deep networks against shallow ML models?
- Neural network compression
- Specialized hardware for deep learning
  - Movidius (http://www.movidius.com/) - deep learning accelerator

More?

Where to look?

- One-stop shop
  – https://github.com/ChristosChristofidis/awesome-deep-learning
- Check this out for hours of fun and amazement
  – http://fastml.com/deep-nets-generating-stuff/
- Books (on Deep Learning)
  – http://www.deeplearningbook.org
  – http://neuralnetworksanddeeplearning.com/
- Programming
  – Theano/Pylearn2, Caffe, Torch (used in Google, Facebook and other companies), TensorFlow – recent initiative of Google, Keras