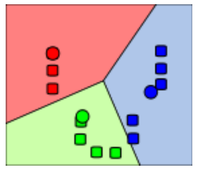
From (chemical) data to information

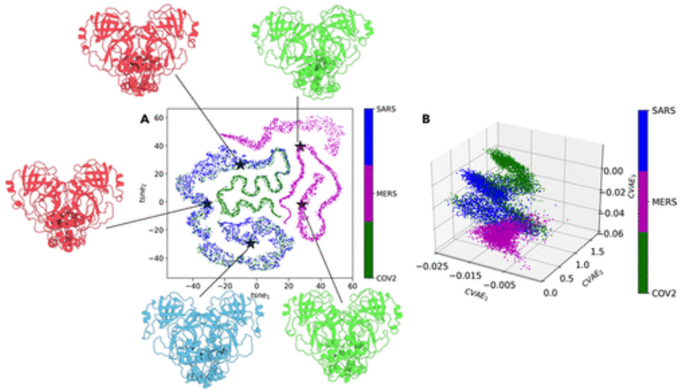
Introduction to Machine Learning Lecture 4

Dr Antonia (Toni) Mey

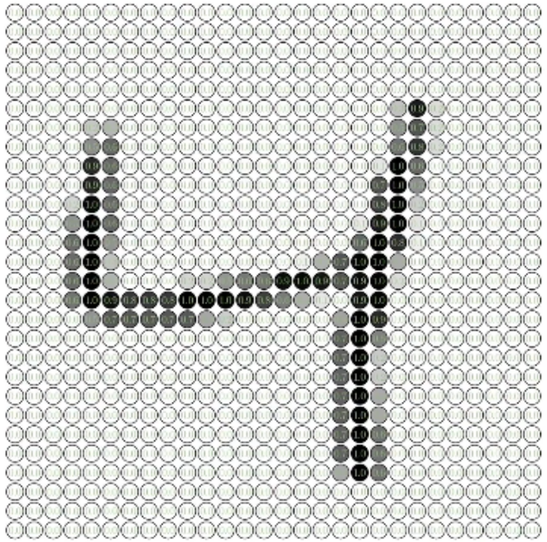
antonia.mey@ed.ac.uk

Room B23 JBB



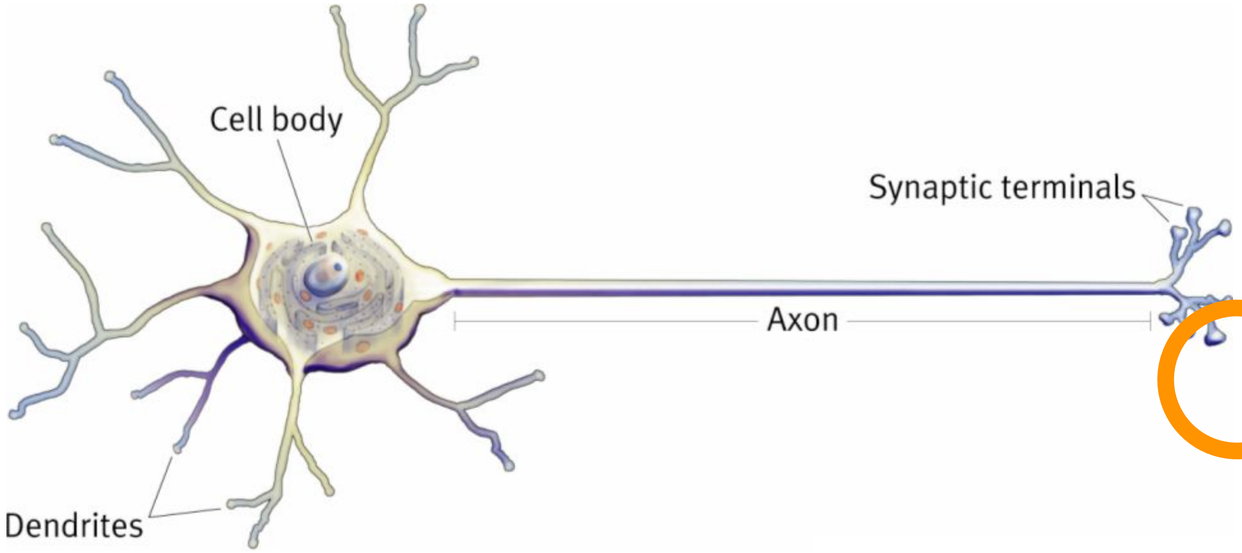


We can recognise different patterns from experience

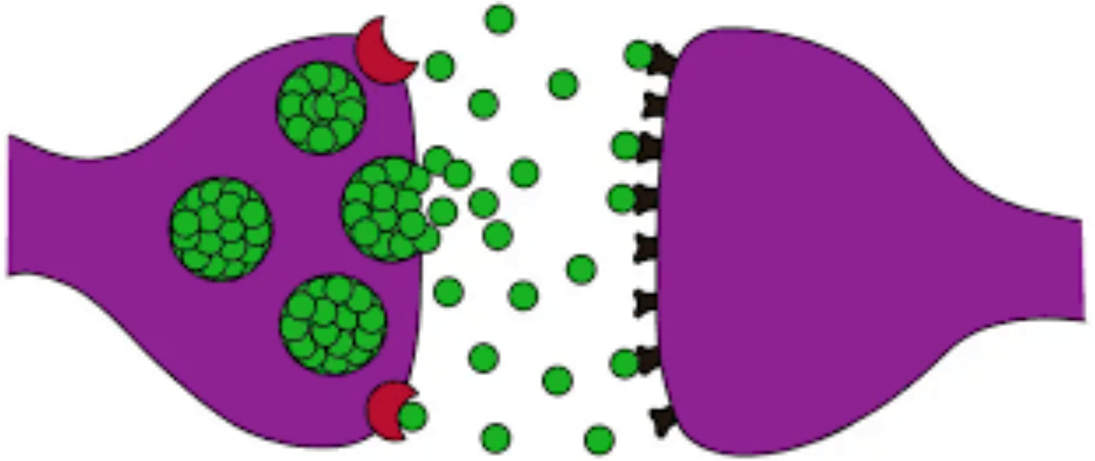


4

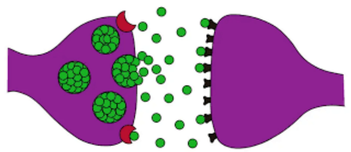
Artificial Neural Network

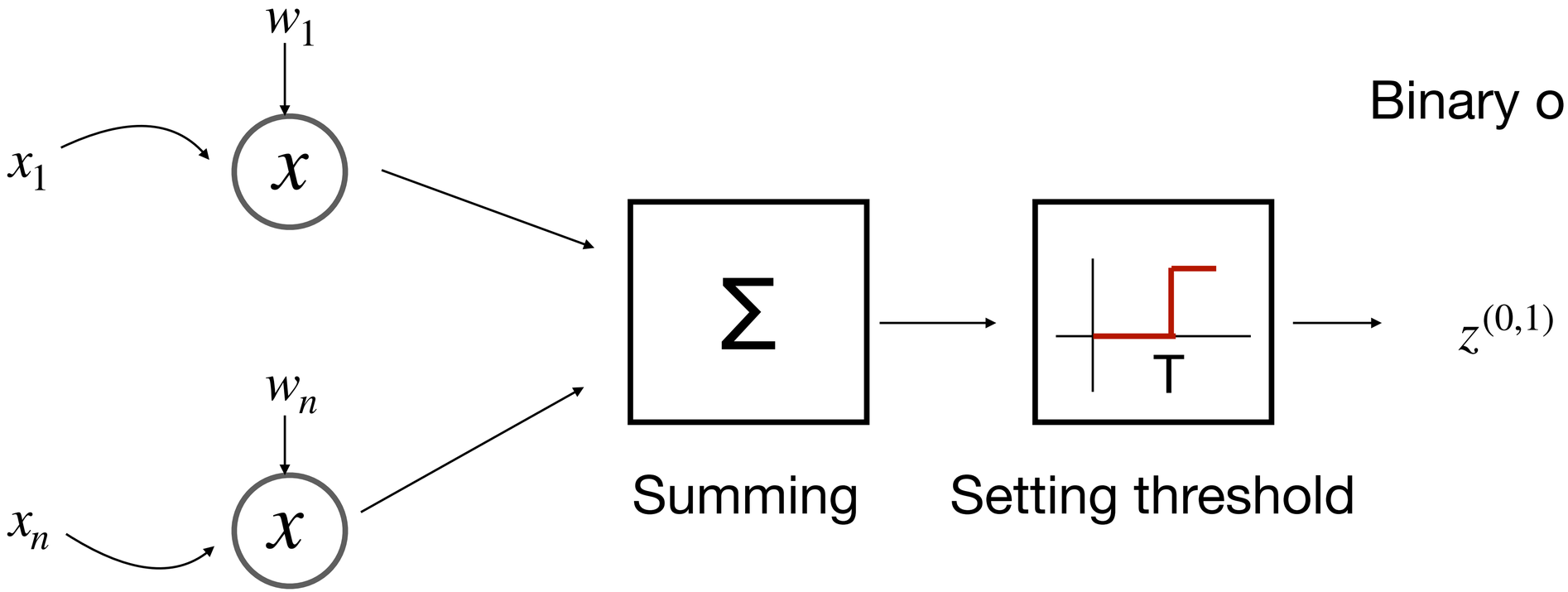


A **neuron** fires if input signal is above a threshold



What happens when a neuron fires?

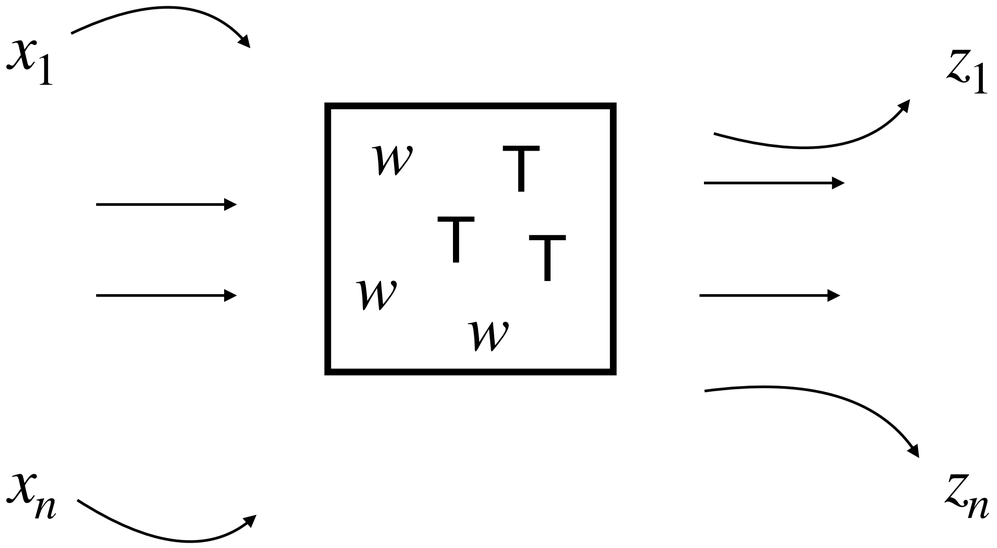




A simplified brain

$$\bar{z} = f(\bar{x}, \bar{w}, \bar{T})$$

$$\bar{d} = g(\bar{x})$$



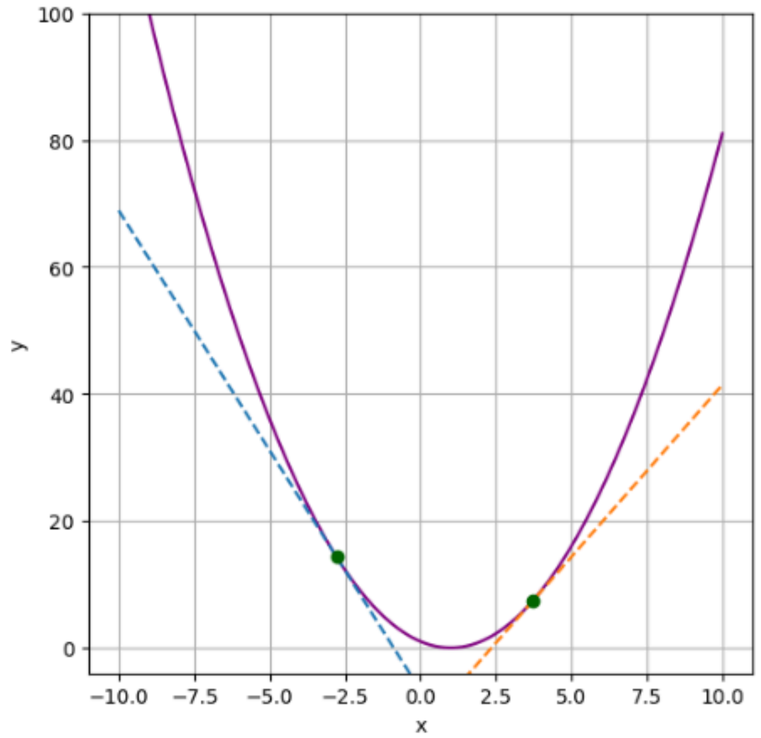
Inputs

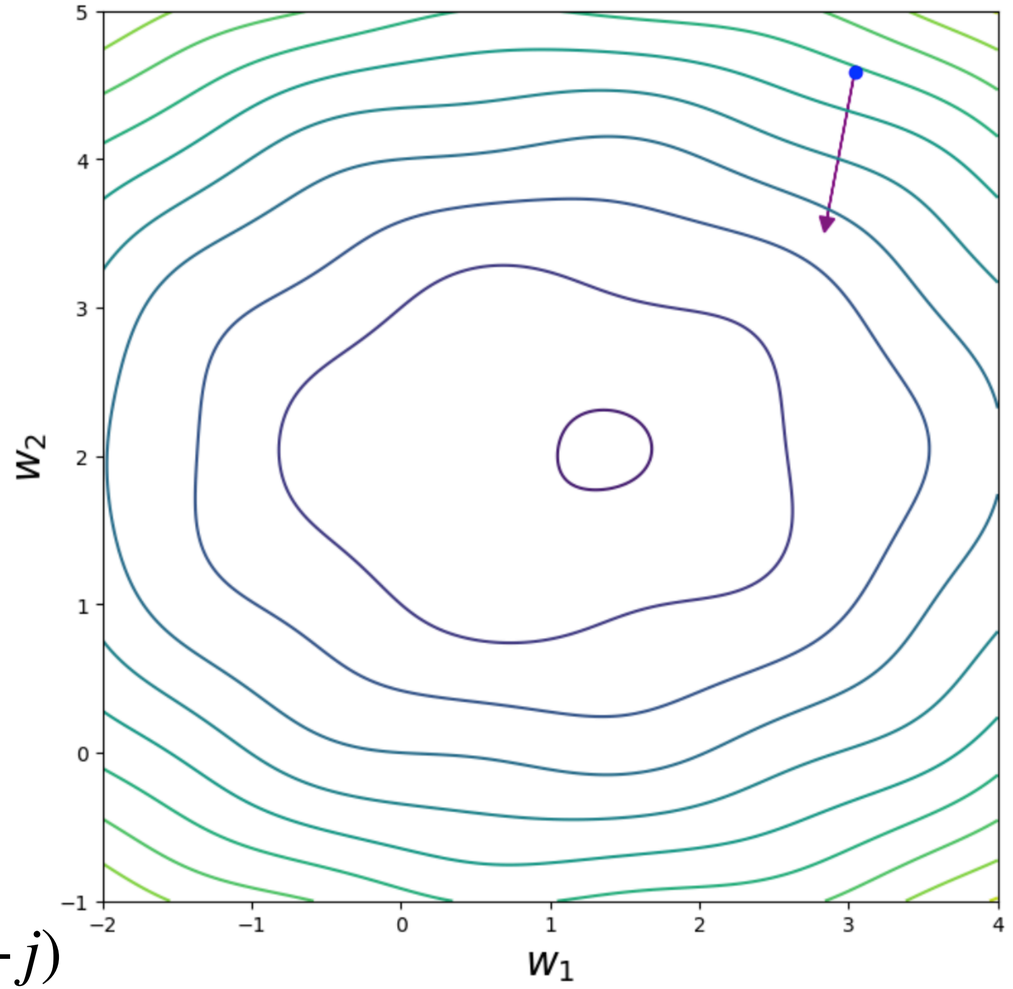
Outputs

We can measure the difference $||\bar{d} - \bar{z}||$

Finding the best weights

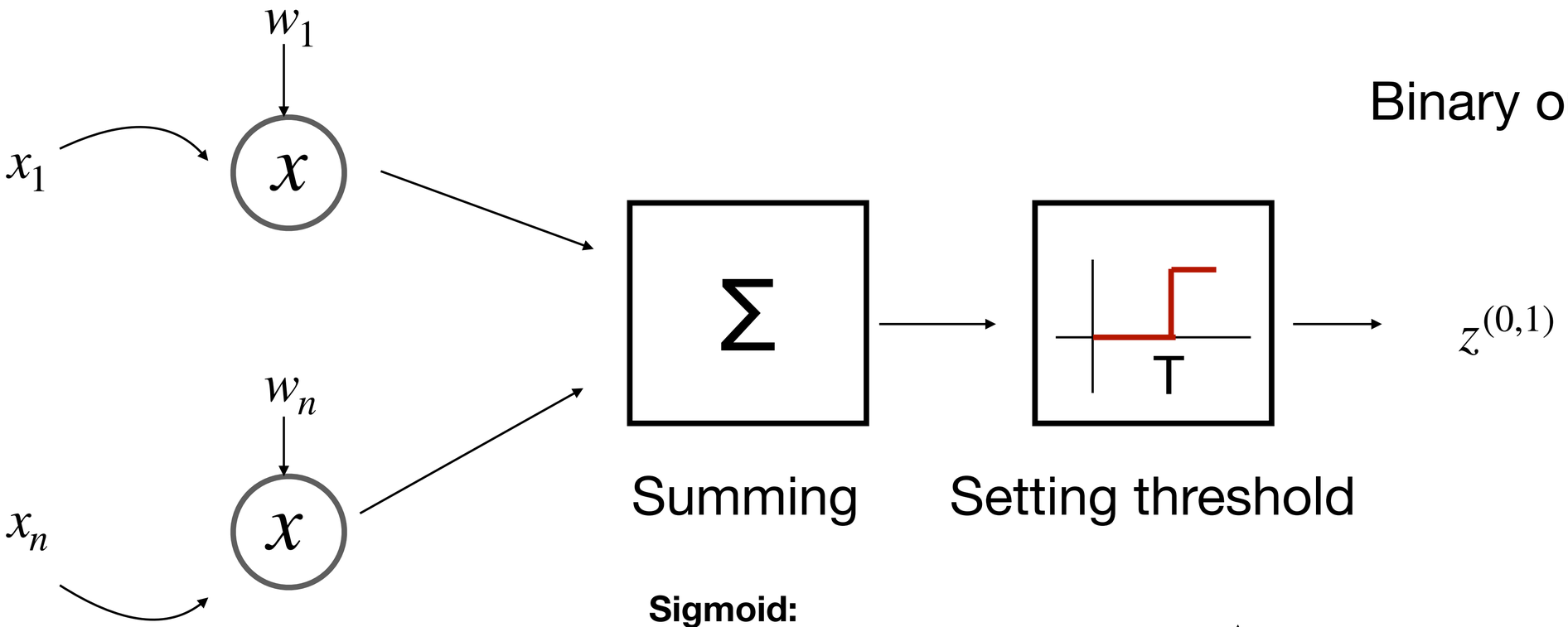
We can measure the difference $P = ||\bar{d} - \bar{z}||^2$





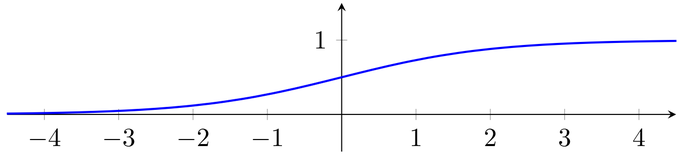
$$\Delta\bar{w} = \frac{\partial P}{\partial w_1} i + \frac{\partial P}{\partial w_2} j \qquad \Delta\bar{w} = r(\frac{\partial P}{\partial w_1} i + \frac{\partial P}{\partial w_2} j)$$

The simplest neuron can learn



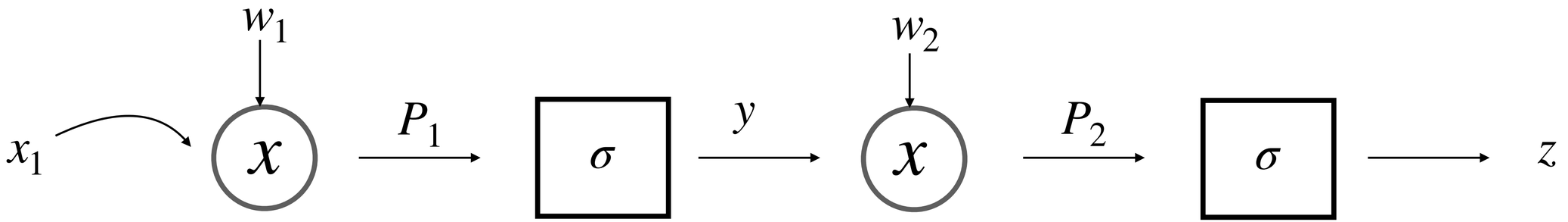
Sigmoid:

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



$$\frac{d}{dz}\sigma(z) = \sigma(z) \cdot (1 - \sigma(z))$$

The simplest neuron can learn



$$P = \frac{1}{2}(d - z)^2$$

$$\frac{\partial P}{\partial w_2} = \frac{\partial P}{\partial z}\frac{\partial z}{\partial w_2} \quad \frac{\partial z}{\partial P_2}\frac{\partial P_2}{\partial w_2}$$

$$\frac{\partial P}{\partial w_1} = \ldots.$$

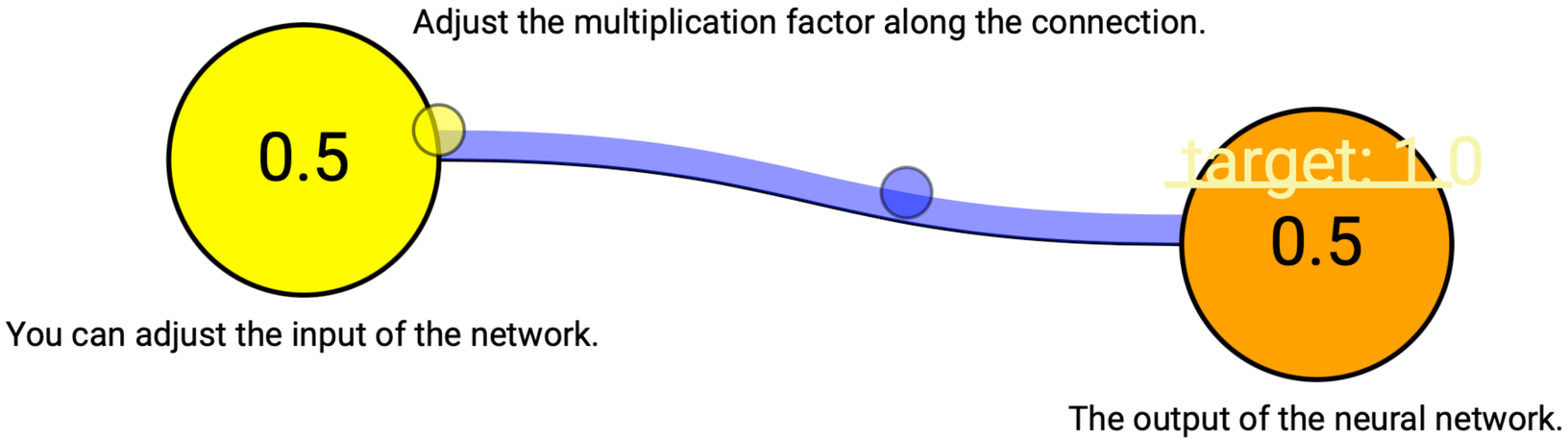
y $(d - z)$

$$\frac{\partial P_2}{\partial w_2} \quad \frac{\partial z}{\partial P_2} \quad \frac{\partial P}{\partial z}$$

$$\sigma(z) \cdot (1 - \sigma(z))$$

Simple networks lets you intuitively play with weights

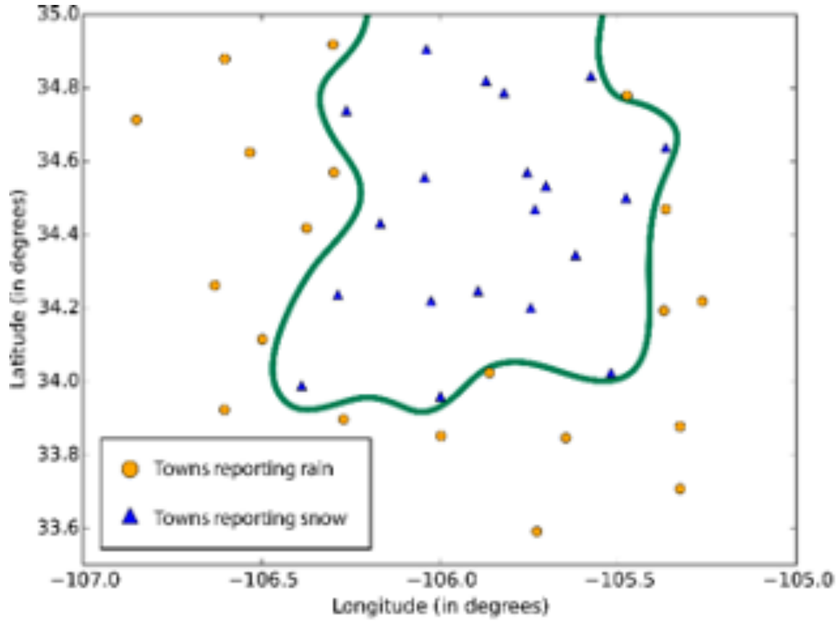
A simple Network: Double its input!

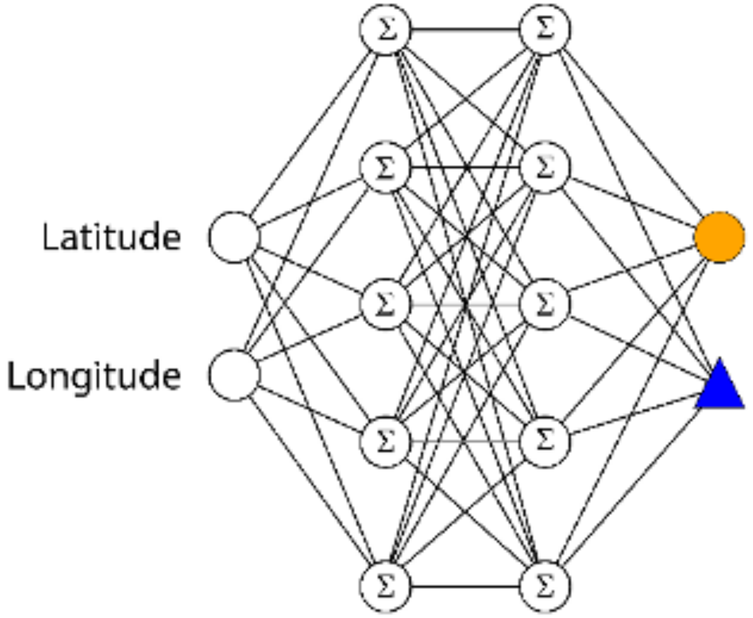


https://imaginary.github.io/simple-networks/index.html#1

Artificial Neural Network

Neurons can be arranged in **networks**
Hidden layers enable producing any complex boundary (in classification) or data fitting model (in regression)





Training the ANN = finding synapses weights w_i minimizing error

ANN can fit any data, but are not easily interpretable

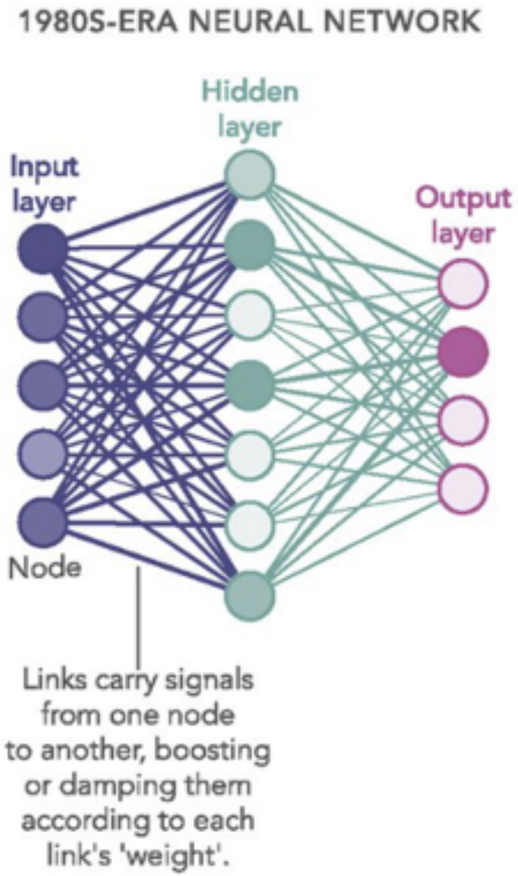
Neural networks can approximate any function

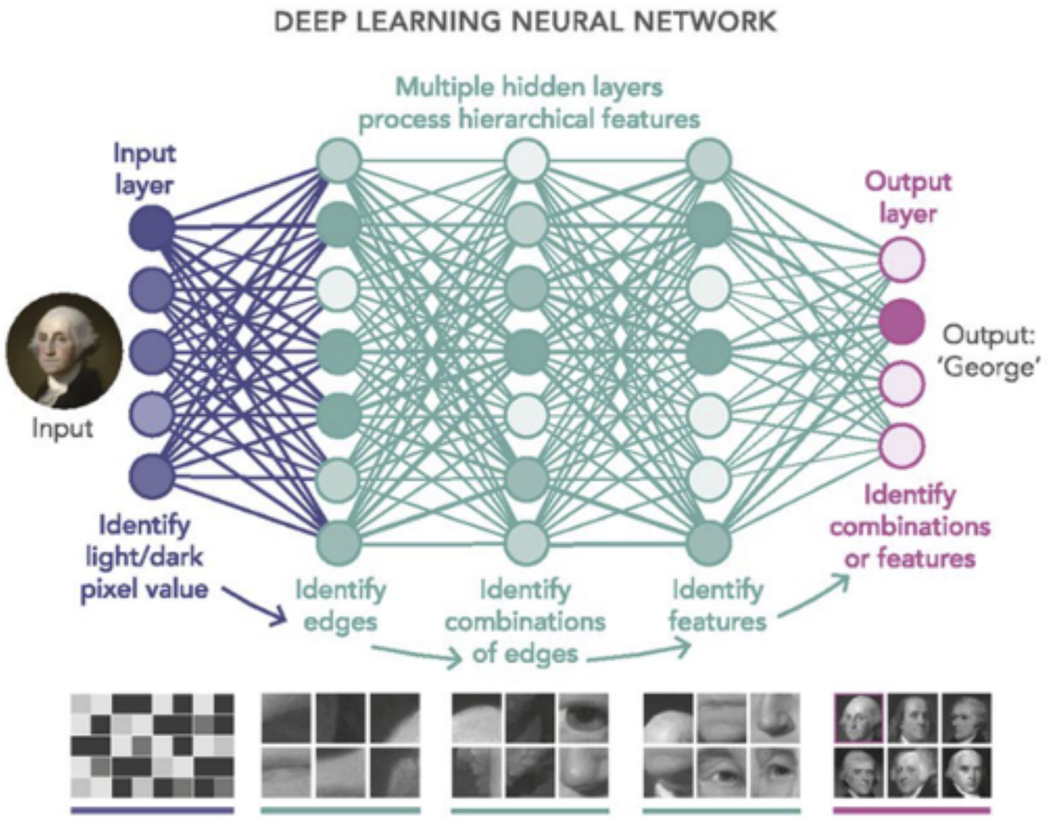
A Neural Network with one hidden layer can represent:

- any bounded continuous function to a given error ε
- any boolean function can be approximated exactly

This is known as the universal approximation theorem and was shown by George Cybenko in 1989.

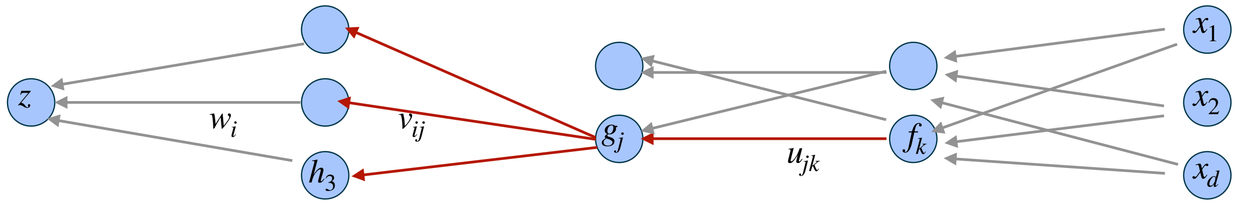
Multilayer perceptron





M.M. Waldrop, *News Feature: What are the limits of deep learning?*, ***PNAS***, 2019

A neural network is trained with back propagation



1. Receive new observation $x = [x_1, \ldots, x_d]$ and known output $z*$

2. Feed forward: compute each unit in hidden layer from information on the previous layer $g_j = \sigma(u_{j0} + \sum_k u_{jk} f_k)$

3. Predict the error $z - z*$

4. Backpropagate to update the weights based on the gradients.

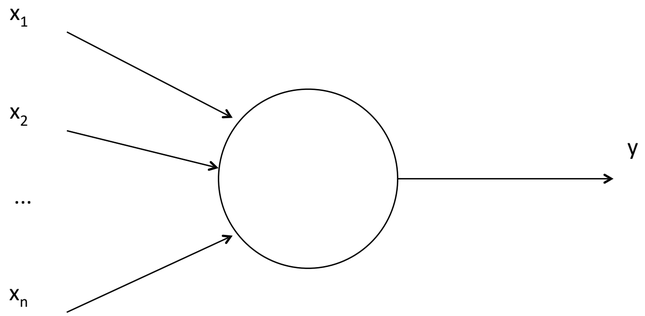
Multilayer perceptron: the maths

Weighted input/output Weighted matrix input Bias

$$z^{(1)} = W^{(0)} a^{(0)} + b^{(1)}$$

$$a^{(l)} = \varphi^{(l)}(z^{(l)}) = \varphi^{(l)}\left(W^{(l-1)} a^{(l-1)} + b^{(l)}\right).$$

Activation function

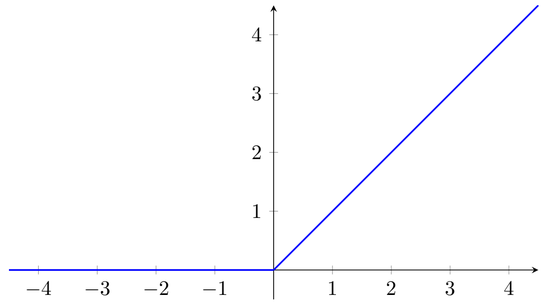


$$f(a_1w_1 + a_2w_2 + \ldots + a_nw_n + b) = z$$

Activation functions:

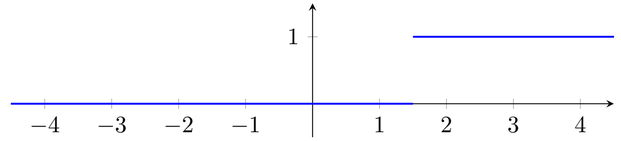
ReLu:

$$\mathrm{ReLU}(z) = \begin{cases} 0, & z \leq 0 \\ z, & z > 0 \end{cases}$$



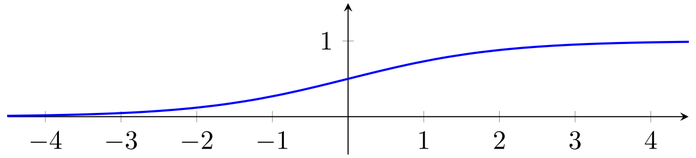
Binary Threshold:

$$s_t(z) = \begin{cases} 0, & z \leq t \\ 1, & z > t \end{cases}$$



Sigmoid:

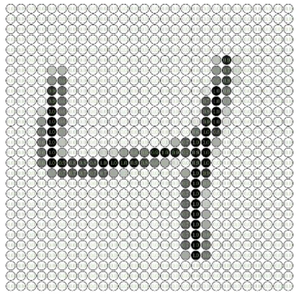
$$\mathrm{sig}(z) = \frac{1}{1 + e^{-z}}$$

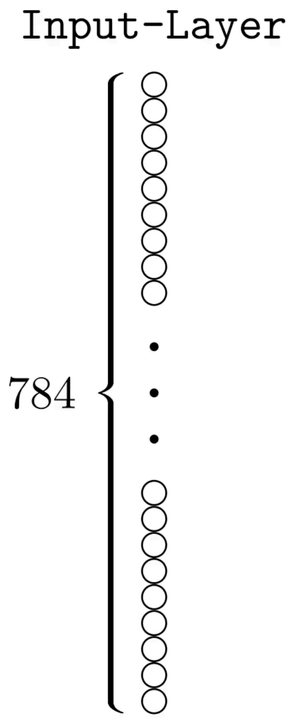


Softmax:

$$\sigma(z) = \frac{1}{\sum_{j=0}^{k-1} e^{z_j}} \begin{pmatrix} e^{z_0} \\ e^{z_1} \\ \vdots \\ e^{z_{k-1}} \end{pmatrix}$$

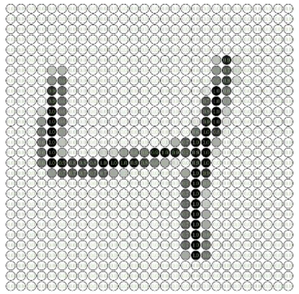
Multiple Output classification using a Multilayer Perceptron

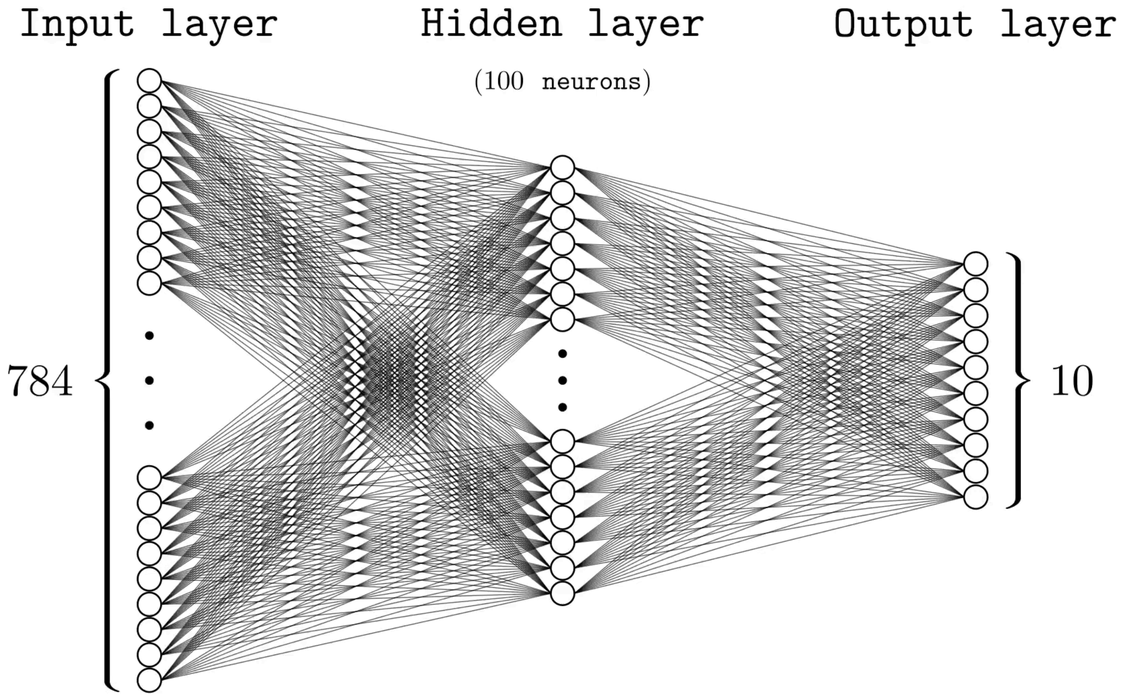




4

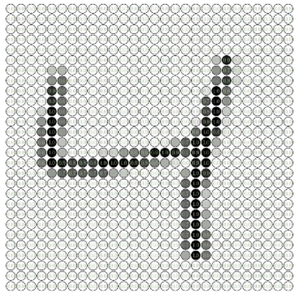
Multiple Output classification using a Multilayer Perceptron





4

Multiple Output classification using a Multilayer Perceptron



Artificial Neural Network (ANN)

4

https://www.i-am.ai/neural-numbers.html

Valuable lessons on machine learning

https://xkcd.com/1838/