

# COMP9414/9814/3411: Artificial Intelligence

## 12. Perceptrons

Russell & Norvig: 18.6, 18.7

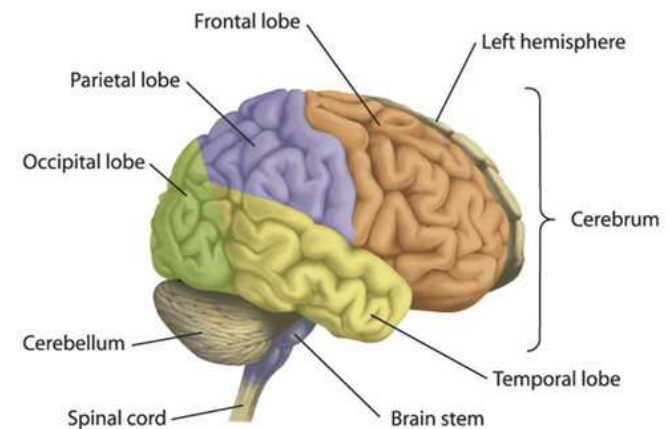
## Outline

- Neurons – Biological and Artificial
- Perceptron Learning
- Linear Separability
- Multi-Layer Networks

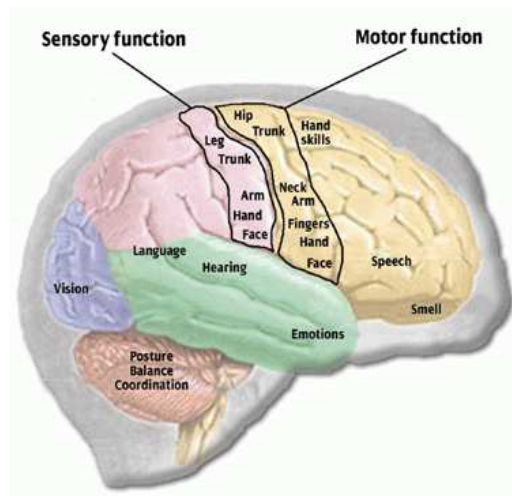
## Sub-Symbolic Processing



## Brain Regions



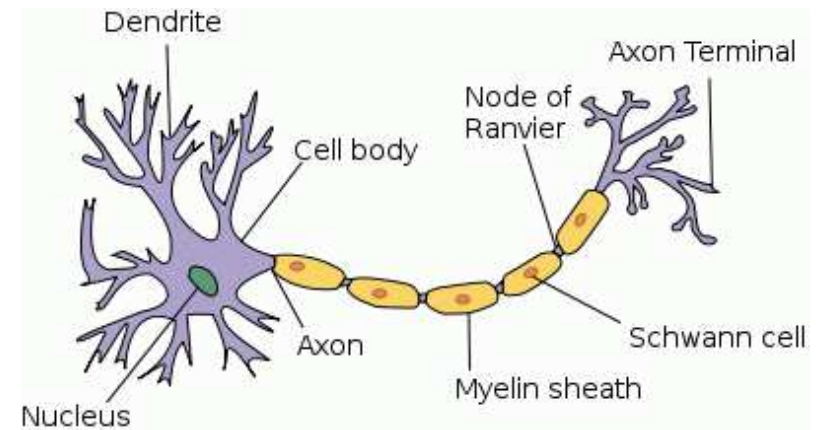
## Brain Functions



UNSW

©Alan Blair, 2013-16

## Structure of a Typical Neuron



UNSW

©Alan Blair, 2013-16

## Biological Neurons

The brain is made up of **neurons** (nerve cells) which have

- a cell body (soma)
- **dendrites** (inputs)
- an **axon** (outputs)
- **synapses** (connections between cells)

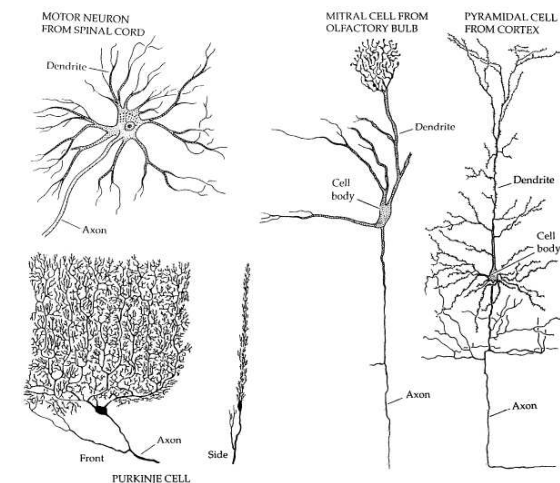
Synapses can be **excitatory** or **inhibitory** and may change over time.

When the inputs reach some threshold an **action potential** (electrical pulse) is sent along the axon to the outputs.

UNSW

©Alan Blair, 2013-16

## Variety of Neuron Types



UNSW

©Alan Blair, 2013-16

## The Big Picture

- human brain has 100 billion neurons with an average of 10,000 synapses each
- latency is about 3-6 milliseconds
- therefore, at most a few hundred “steps” in any mental computation, but massively parallel

## Artificial Neural Networks

(Artificial) Neural Networks are made up of nodes which have

- inputs edges, each with some **weight**
- outputs edges (with **weights**)
- an **activation level** (a function of the inputs)

Weights can be positive or negative and may change over time (learning).

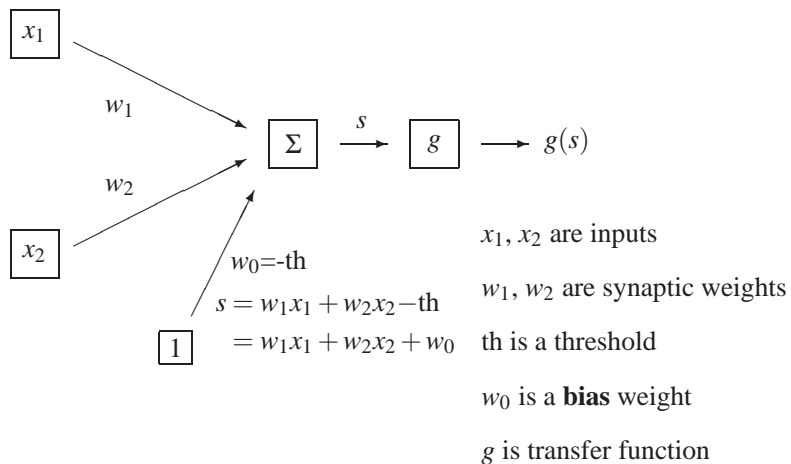
The **input function** is the weighted sum of the activation levels of inputs.

The activation level is a non-linear **transfer** function  $g$  of this input:

$$\text{activation}_i = g(s_i) = g\left(\sum_j w_{ij}x_j\right)$$

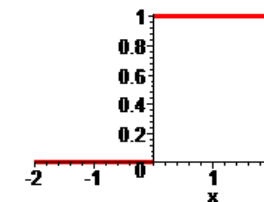
Some nodes are inputs (sensing), some are outputs (action)

## Rosenblatt Perceptron



## Transfer function

Originally, a (discontinuous) step function was used for the transfer function:



$$g(s) = \begin{cases} 1, & \text{if } s \geq 0 \\ 0, & \text{if } s < 0 \end{cases}$$

(Later, other transfer functions were introduced, which are continuous and smooth)

## Linear Separability

Q: what kind of functions can a perceptron compute?

A: linearly separable functions

Examples include:

AND  $w_1 = w_2 = 1.0, w_0 = -1.5$

OR  $w_1 = w_2 = 1.0, w_0 = -0.5$

NOR  $w_1 = w_2 = -1.0, w_0 = 0.5$

Q: How can we train it to learn a new function?

## Perceptron Learning Rule

Adjust the weights as each input is presented.

recall:  $s = w_1 x_1 + w_2 x_2 + w_0$

if  $g(s) = 0$  but should be 1,      if  $g(s) = 1$  but should be 0,

$$w_k \leftarrow w_k + \eta x_k$$

$$w_k \leftarrow w_k - \eta x_k$$

$$w_0 \leftarrow w_0 + \eta$$

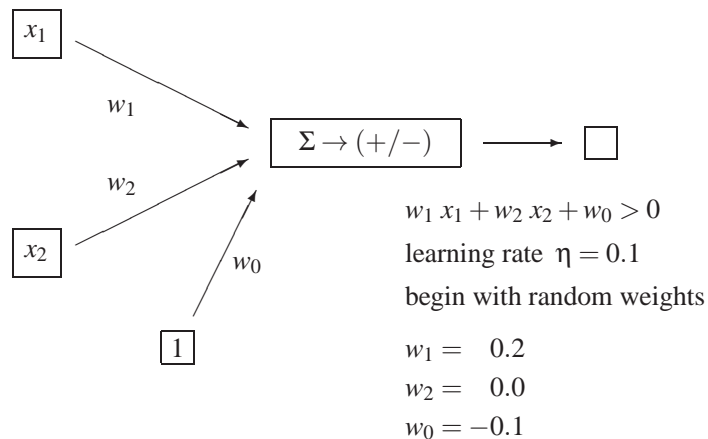
$$w_0 \leftarrow w_0 - \eta$$

$$\text{so } s \leftarrow s + \eta \left(1 + \sum_k x_k^2\right) \quad \text{so } s \leftarrow s - \eta \left(1 + \sum_k x_k^2\right)$$

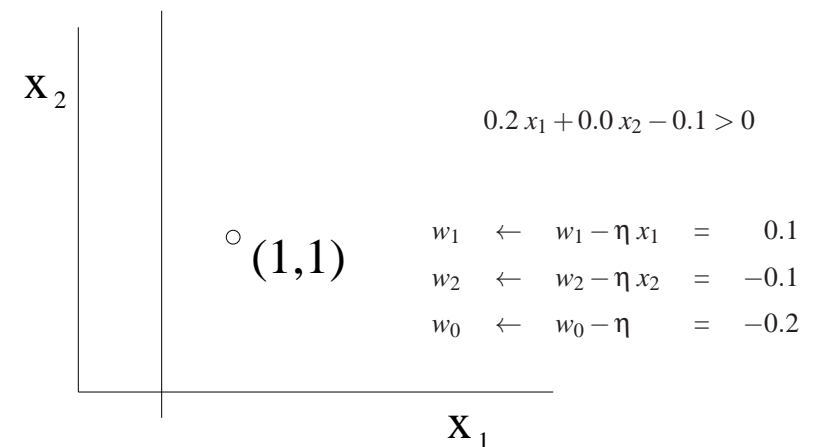
otherwise, weights are unchanged. ( $\eta > 0$  is called the **learning rate**)

**Theorem:** This will eventually learn to classify the data correctly, as long as they are **linearly separable**.

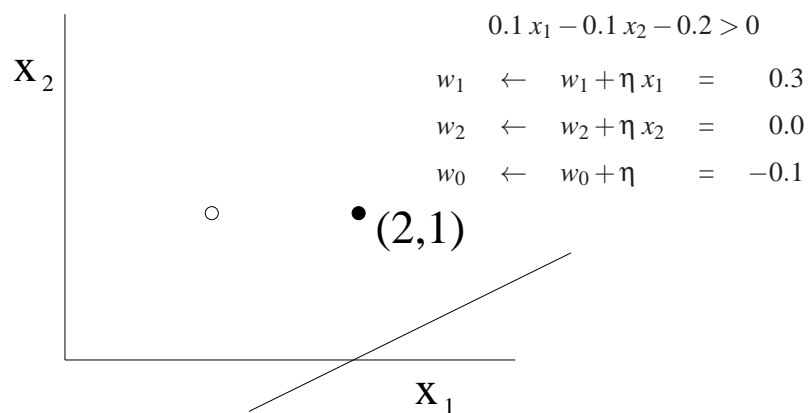
## Perceptron Learning Example



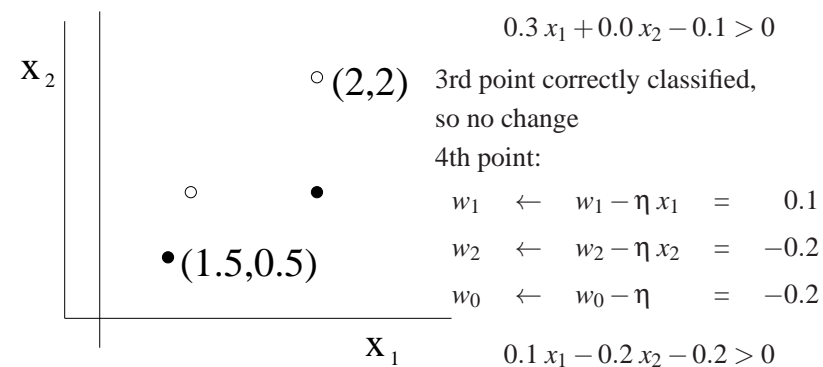
## Training Step 1



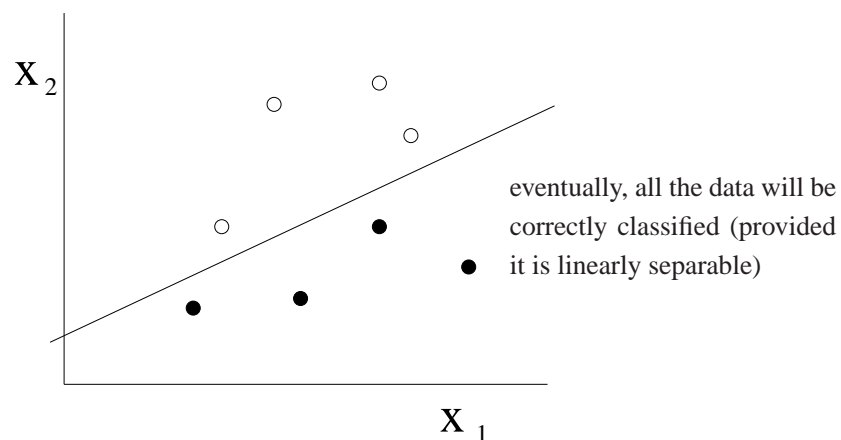
## Training Step 2



## Training Step 3

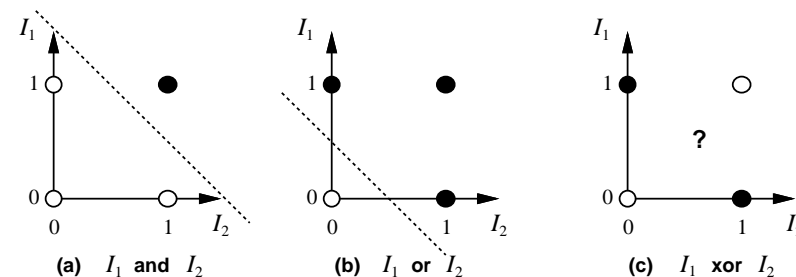


## Final Outcome



## Limitations

Problem: many useful functions are not linearly separable (e.g. XOR)



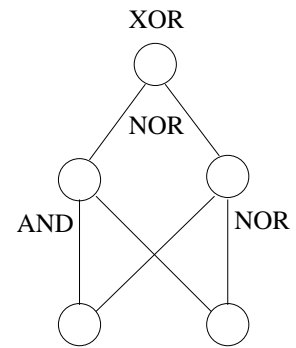
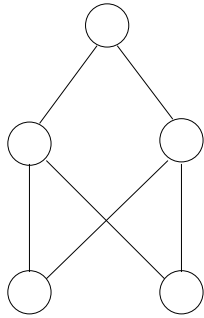
Possible solution:

$x_1 \text{ XOR } x_2$  can be written as:  $(x_1 \text{ AND } x_2) \text{ NOR } (x_1 \text{ NOR } x_2)$

Recall that AND, OR and NOR can be implemented by perceptrons.

## Multi-Layer Neural Networks

---



Problem: How can we train it to learn a new function? (credit assignment)

[stay tuned...]