Introduction to Artificial Intelligence

COMP307/AIML420

Neural Networks 1: The Perceptron

Dr Andrew Lensen

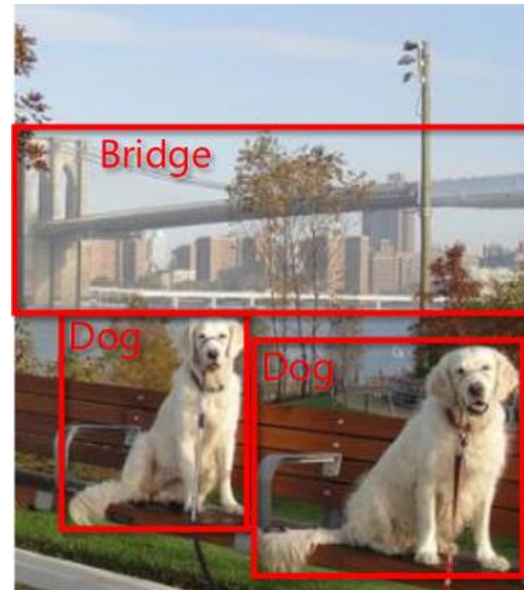
Andrew.Lensen@vuw.ac.nz

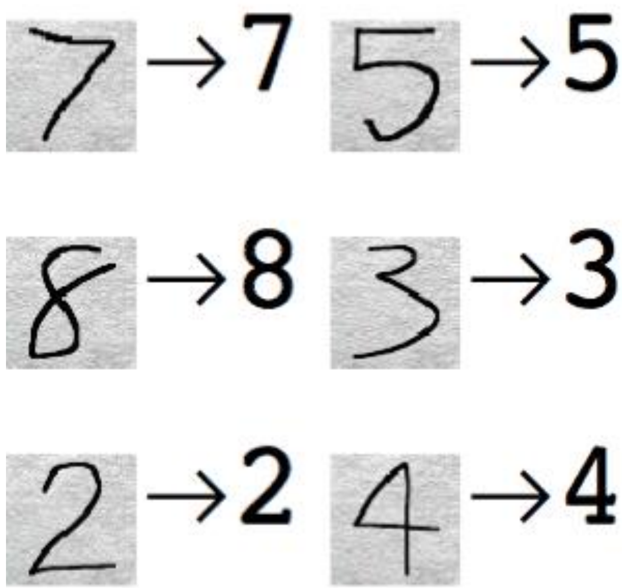
Outline

- Why ANN?
- Origin
- The perceptron
- Perceptron learning
- What can (not) perceptron learn
- Extending perceptron

Why Artificial Neural Networks (ANNs)?

- “Killer” applications in a lot of areas
  - Computer vision/image processing
  - Playing games (AlphaGo, Watson, …)
  - Big data
  - …





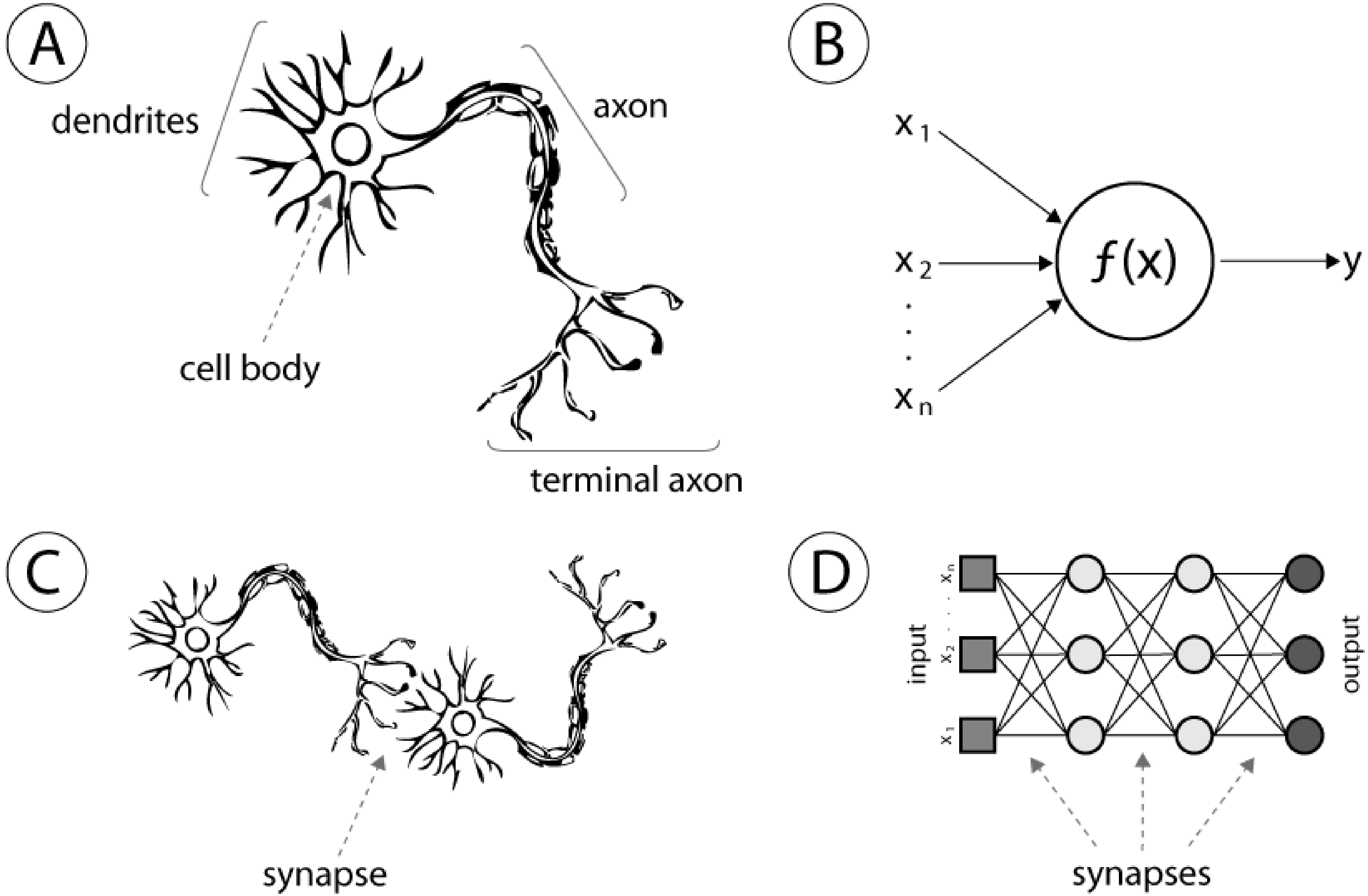
Origin

- Human brain shows amazing capability in:
  - Learning
  - Perception
  - Adaptability
  - …

- Can we *simulate the* human brain to achieve the above functionalities?

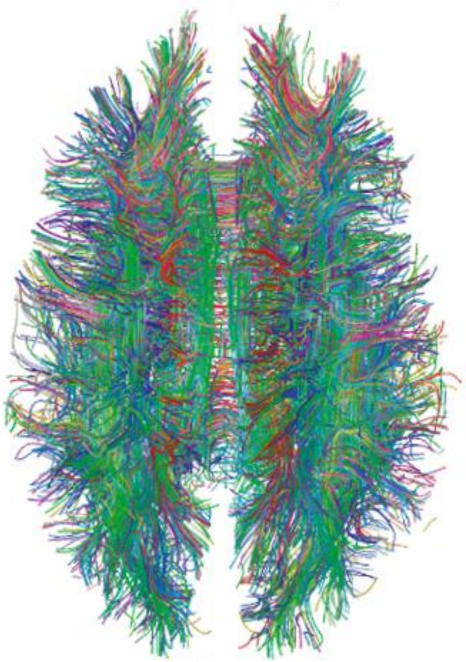
- **Artificial** neural networks use this as inspiration/motivation

Origin

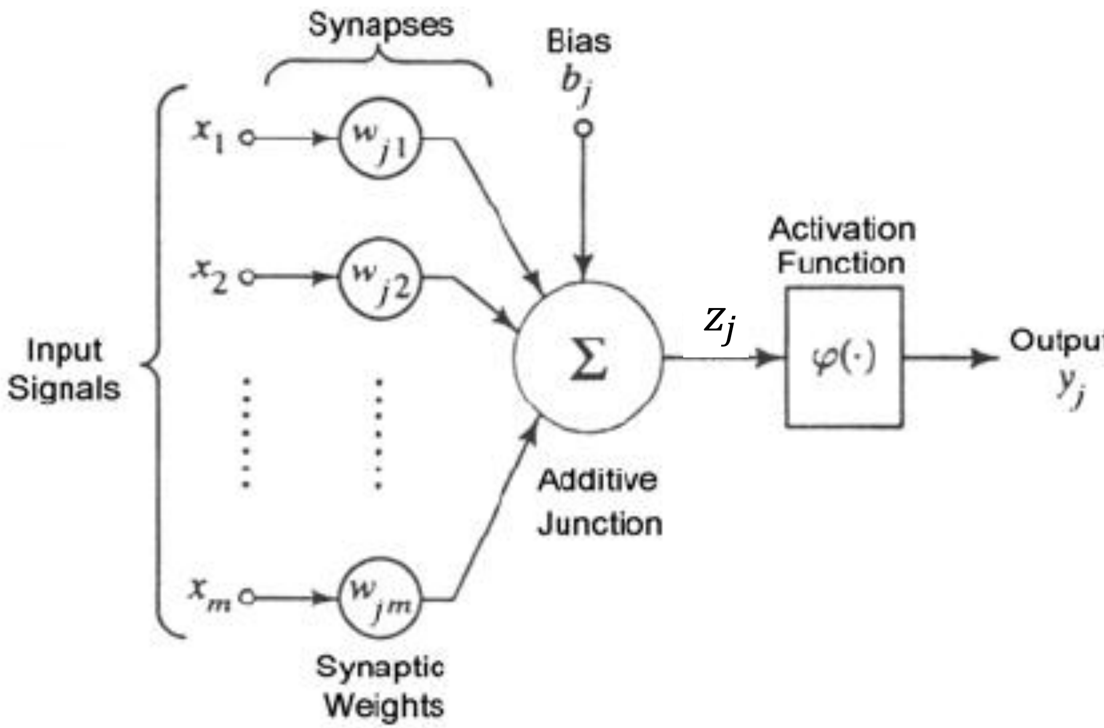

A
dendrites
axon
cell body
terminal axon
B
x_1
x_2
x_n
$f(x)$
y_i
C
synapse
D
input
x_1
x_2
x_n
output
synapses

Origin

- Facts about human brain:
  - 10^{11} neurons, massively connected
  - Each neuron is connected to 1000–10,000 other neurons
  - 10^{14} to 10^{15} connections in total!
  - Brain message passing is 1 million times **slower** than modern electronic circuits
  - But very *efficient* for complex decision making
  - Usually fewer than 100 serial stages
  - 100 step rule (500ms response time)



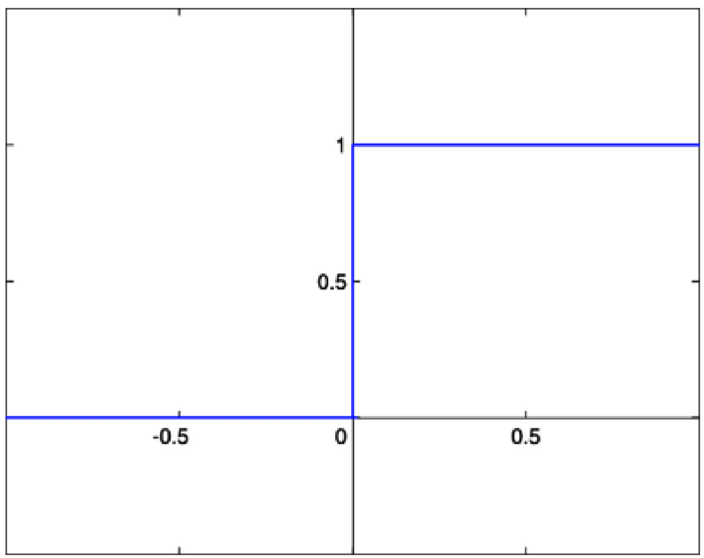
A Single Artificial Neuron

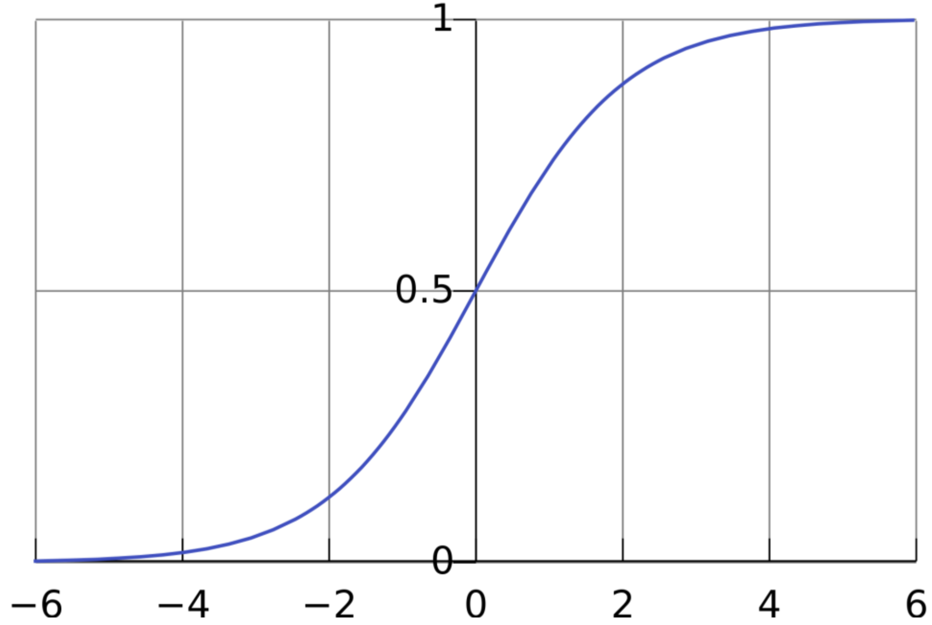


$$z_j = \sum_{i=1}^{m} w_{ji} x_i + b_j$$

$$y_j = \varphi(z_j)$$

Activation Functions





Threshold

$$y_j = \begin{cases} 1, & \text{if } z_j > 0, \\ 0, & \text{otherwise} \end{cases}$$

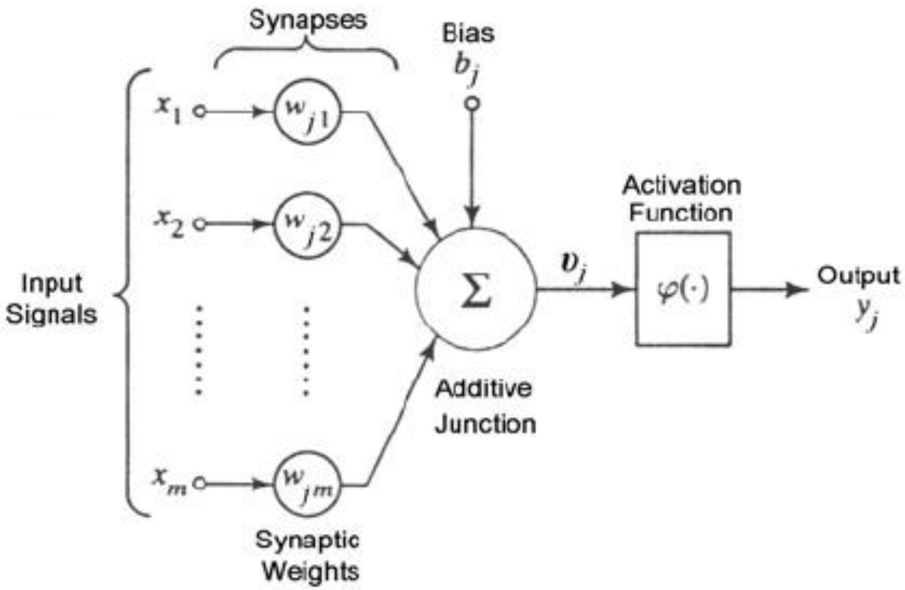
Sigmoid

$$y_j = \frac{1}{1 + e^{-\beta z_j}}$$

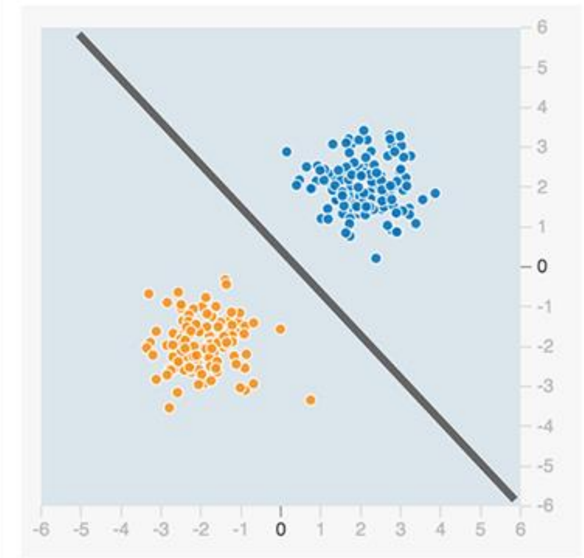
$\boxed{\textit{Why do we need this?}}$

Perceptron

- A *special* type of artificial neuron
  - Real-valued inputs
  - Binary output
  - Threshold activation function



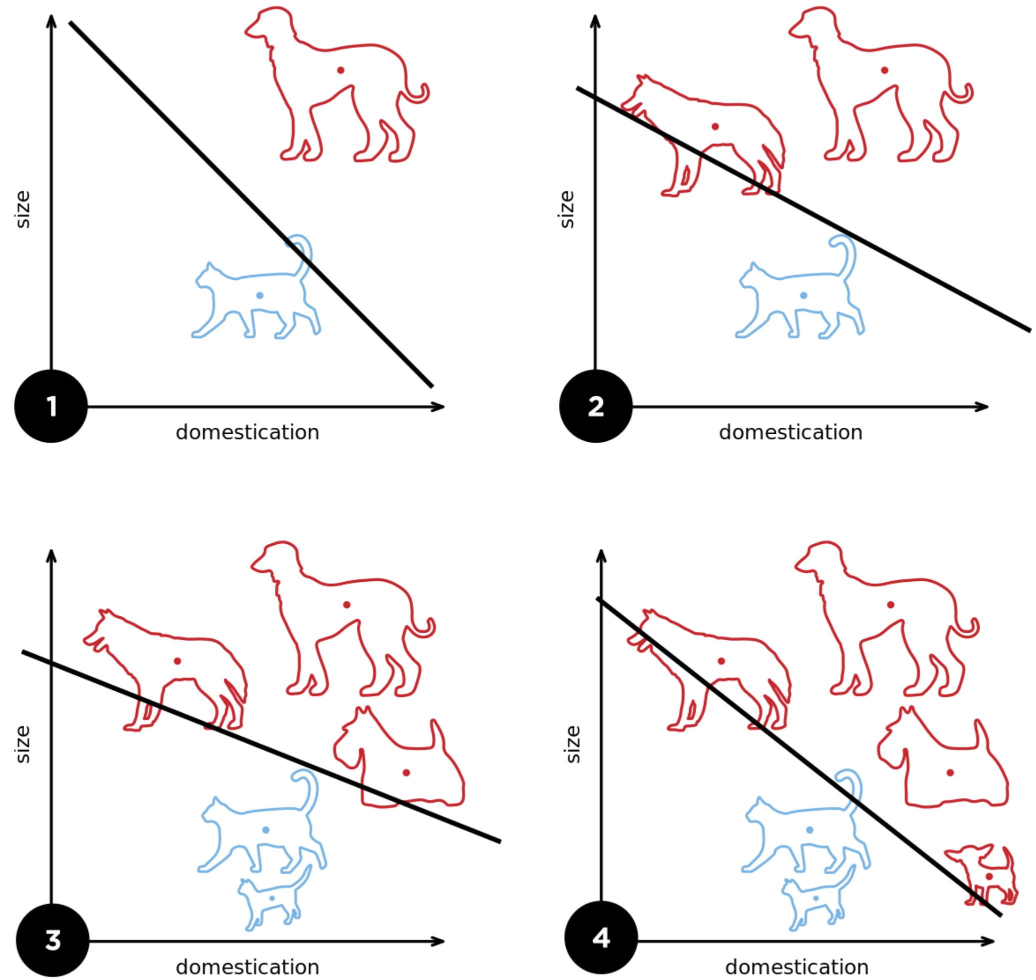
$$y_j = \begin{cases} 1, & \text{if } \sum_{i=1}^{m} w_{ji} x_i + b_j > 0, \\ 0, & \text{otherwise} \end{cases}$$



$$y = \begin{cases} 1, \text{if } x_1 + x_2 > 0, \\ 0, \text{otherwise.} \end{cases}$$

Perceptron

- To perform linear classification
  - 2 inputs: a line; 3 inputs: a plane
- Can do online learning
  - Update w_{ji} and b_j based on new examples



Learning A Perceptron

- How to get the **optimal** weights and bias?
- Let us only consider accuracy:
  - *Optimal* if 100% accuracy on training set
  - Can have many optimal solutions…
- To simplify notation, we can transform the bias to a weight $w_{j0} = b_j$, where $x_0 = 1$ always holds

$$y_j = \begin{cases} 1, & \text{if } \sum_{i=1}^{m} w_{ji} x_i + b_j > 0, \\ 0, & \text{otherwise} \end{cases}$$

$$\boxed{b_j = w_{j0} \cdot 1 = w_{j0} x_0}$$

$$y_j = \begin{cases} 1, & \text{if } \sum_{i=0}^{m} w_{ji} x_i > 0, \\ 0, & \text{otherwise} \end{cases}$$

Learning A Perceptron

- Initialise weights and threshold randomly (or set all to zero)
- Given a new example/instance $(x_1, x_2, \dots, x_m, d)$:
  - Input feature vector: $(x_1, x_2, \dots, x_m)$
  - Output (**d**esired class label): d
  - Predicted output: y

$$y = \begin{cases} 1, & \text{if } \sum_{i=0}^{m} w_i x_i > 0, \\ 0, & \text{otherwise} \end{cases}$$

- If $y = 0$ and $d = 1$,
  - increase $b = w_0$, increase w_i for each positive x_i,
    decrease w_i for each negative x_i
- If $y = 1$ and $d = 0$,
  - decrease $b = w_0$, decrease w_i for each positive x_i,
    increase w_i for each negative x_i
- Repeat for each new example until desired result achieved

Learning A Perceptron

- Initialise weights and threshold randomly (or set all to zero)
- Given a new example/instance $(x_1, x_2, \dots, x_m, d)$:
  - Input feature vector: $(x_1, x_2, \dots, x_m)$
  - Output (**d**esired class label): d
  - Predicted output: y

$$\boxed{y = \begin{cases} 1, & \text{if } \sum_{i=0}^{m} w_i x_i > 0, \\ 0, & \text{otherwise} \end{cases}}$$

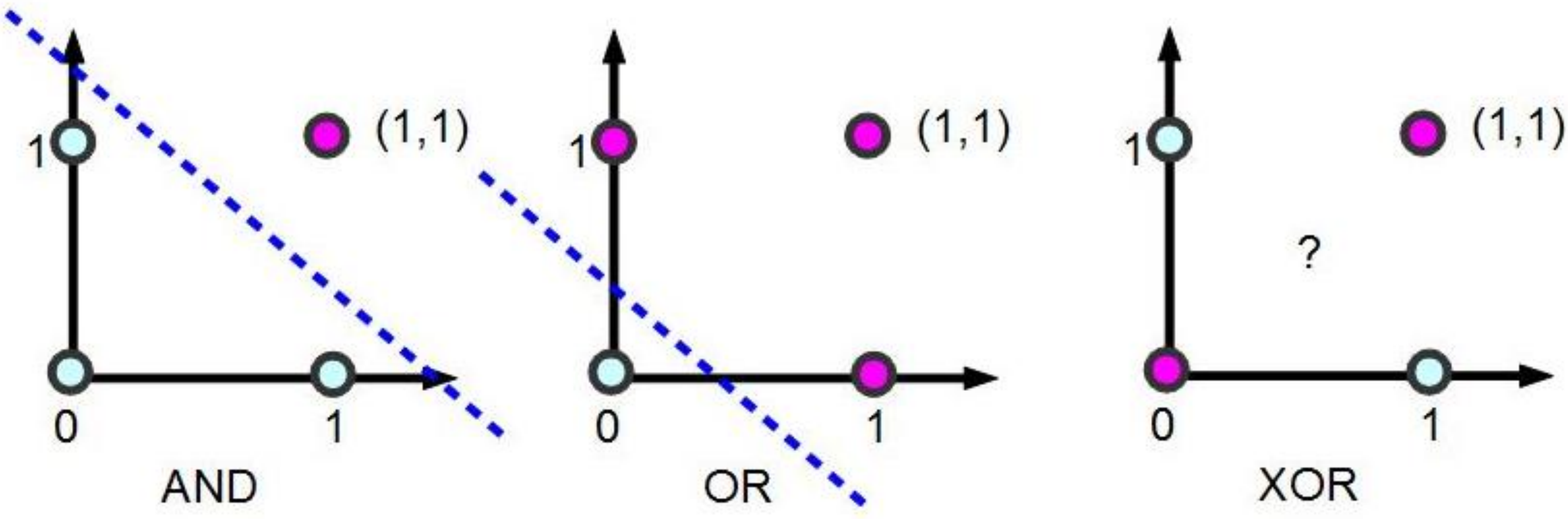
- We can use the equation:

$$\boxed{w_i \leftarrow w_i + \eta(d - y)x_i, i = 0, 1, 2, \dots, m}$$

- Where $\eta \in [0,1]$ is called the learning rate
- Repeat for each new example until desired result achieved

Problem with the Perceptron

- What can the perceptron learn?

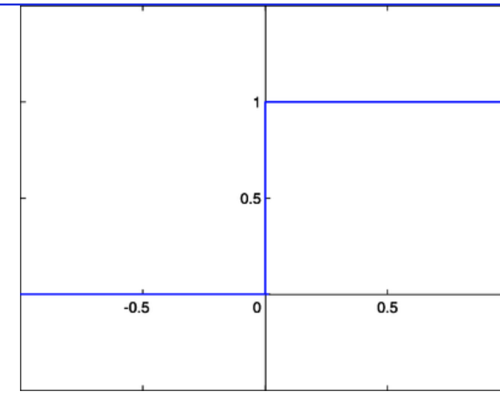


- *Perceptron convergence theorem*: The perceptron learning algorithm will converge **if and only if** the training set is linearly separable.
- Cannot learn for XOR (Minsky and Papert, 1969)

Multi-Layer Perceptron (MLP)

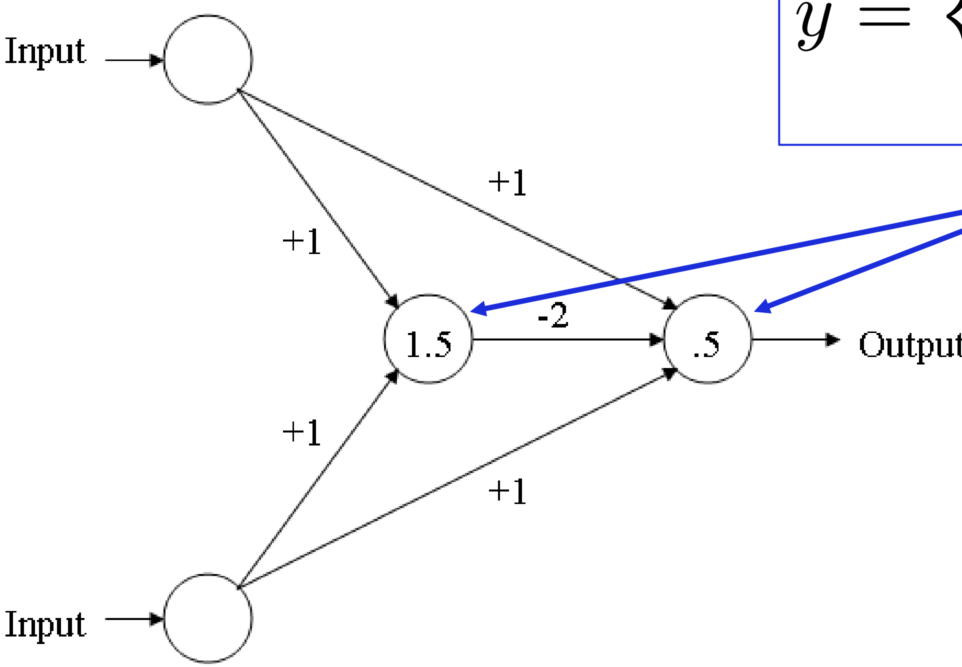
- Add one ***hidden*** node between the inputs and output

x1	x2	y (class)
0	0	0
1	0	1
0	1	1
1	1	0



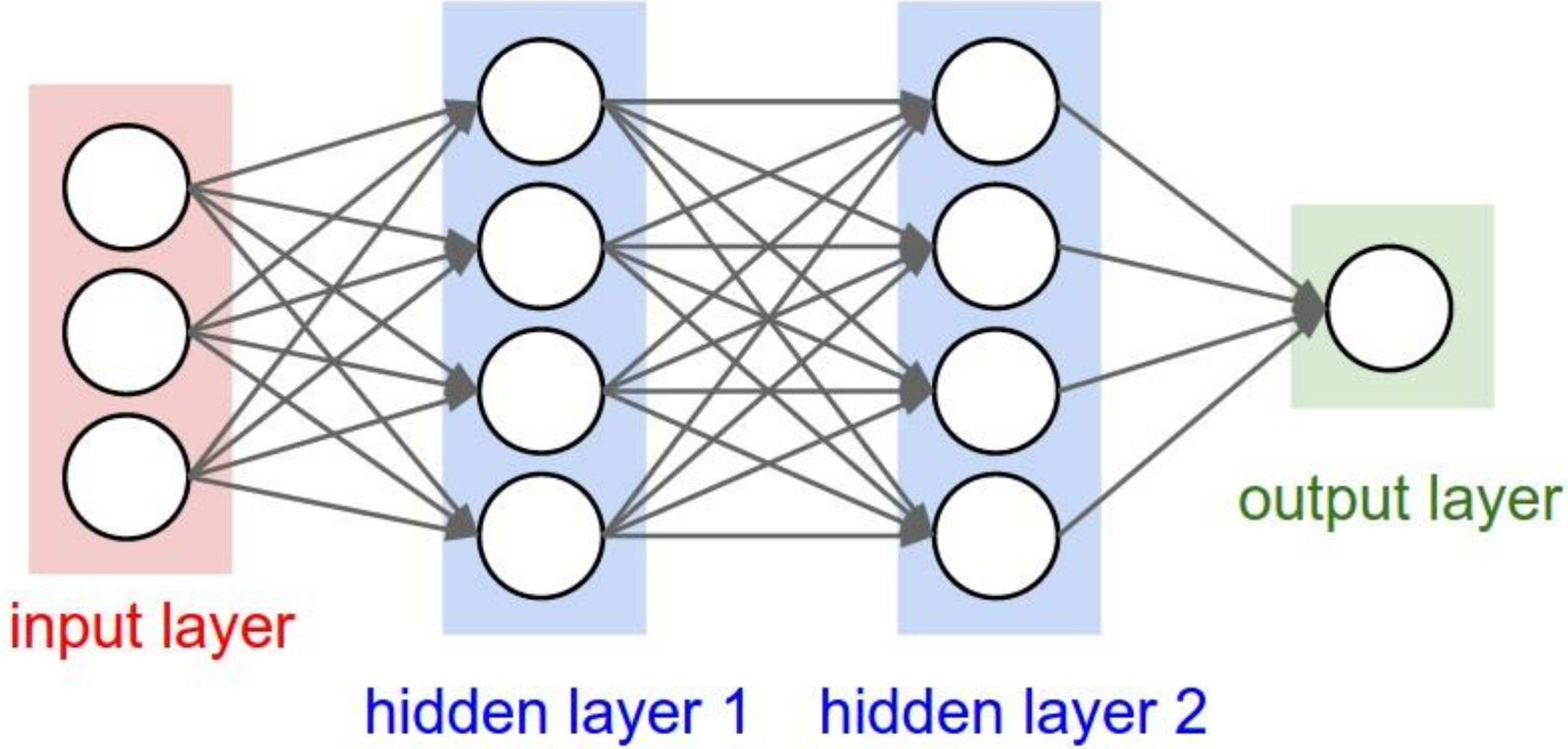
$$y = \begin{cases} 1, & \text{if } \sum_i w_i x_i - T > 0, \\ 0, & \text{otherwise} \end{cases}$$

Threshold



Neural Network

- Add more hidden layers
- Add more nodes for hidden layers



Summary

- ANN has many “killer” applications
  - Image analysis, playing games, …
- Perceptron – the simplest neural network
- How to learn a perceptron
- Limitations of perceptron, multi-layer perceptron, general neural network

- Reading: Text book section 20.5 (2nd edition) or section 18.7 (3rd edition) or Web materials