Welcome to the computational cognitive modelling workshop!

Part 2: Artificial neural networks

Part 2: **Artificial neural networks**

Olivia Guest

Chris Brand

Nick Sexton

Nicole Cruz De Echeverria Loebell

What is a neural network?

A mathematical model

- Inspired by the nervous system
- A set of *units*, connected by *weights*
- The network *runs* by passing *activations* from the *input* (to the *hidden*) to the *output* units

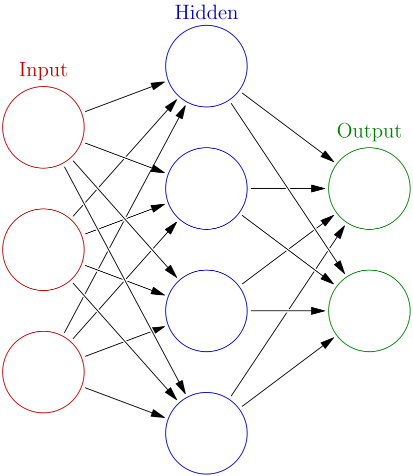


Figure: Glosser.ca / CC-BY-SA-3.0

Why use artificial neural networks for modelling?

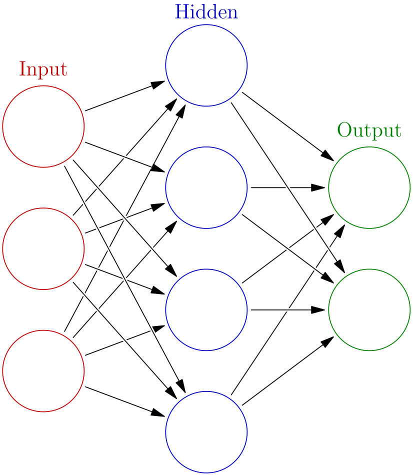
Some aspects of their behaviour are like their namesake!

- Learn pretty much any input-output data
- Uncover rules on their own about data
- Generalise from what they have learnt
- Cope with noise and damage

How does an artificial neural network run?

By using maths, predictably!

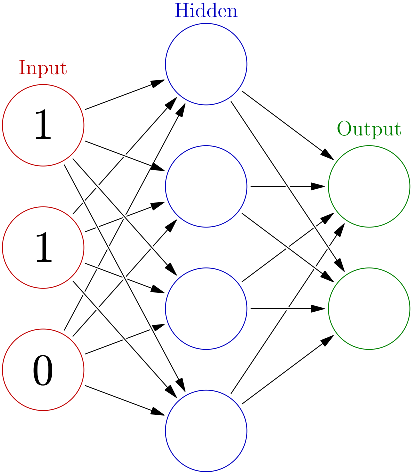
1. Input units are set to a *pattern*



How does an artificial neural network run?

By using maths, predictably!

1. Input units are set to a *pattern*

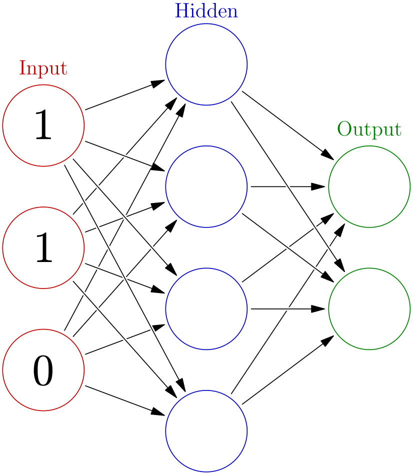


How does an artificial neural network run?

By using maths, predictably!

1. Input units are set to a *pattern*

2. Calculate hidden units' states

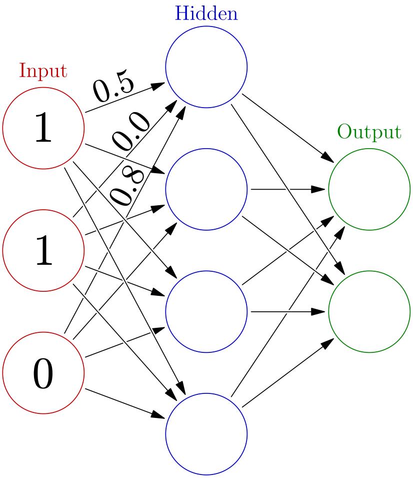


How does an artificial neural network run?

By using maths, predictably!

1. Input units are set to a *pattern*

2. Calculate hidden units' states



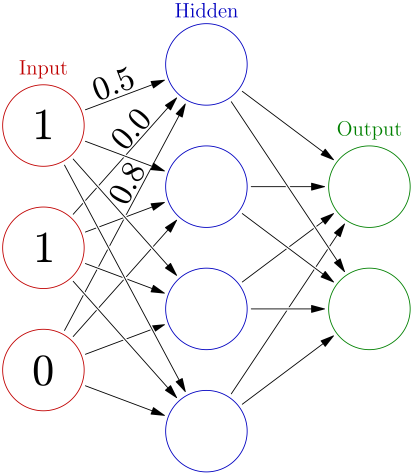
How does an artificial neural network run?

By using maths, predictably!

1. Input units are set to a *pattern*

2. Calculate hidden units' states:

$$1 \times 0.5 \;=\; 0.5$$



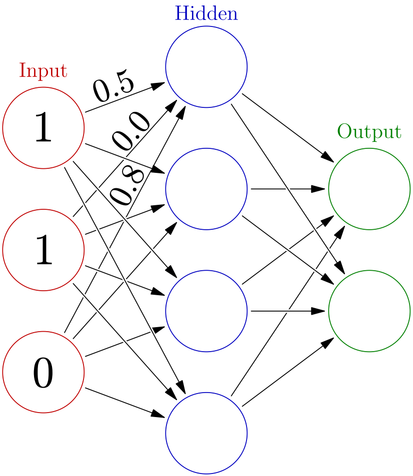
How does an artificial neural network run?

By using maths, predictably!

1. Input units are set to a *pattern*

2. Calculate hidden units' states:

$$\begin{aligned} 1 \times 0.5 &= 0.5 \\ 1 \times 0.0 &= 0.0 \end{aligned}$$



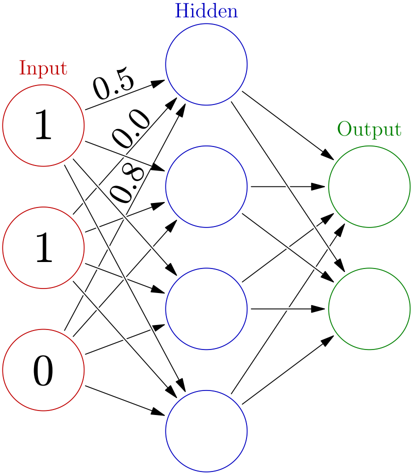
How does an artificial neural network run?

By using maths, predictably!

1. Input units are set to a *pattern*

2. Calculate hidden units' states:

$$
\begin{aligned}
1 \times 0.5 &= 0.5 \\
1 \times 0.0 &= 0.0 \\
0 \times 0.8 &= 0.0
\end{aligned}
$$



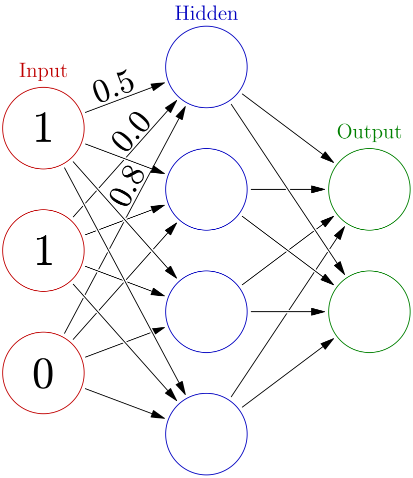
How does an artificial neural network run?

By using maths, predictably!

1. Input units are set to a *pattern*

2. Calculate hidden units' states:

$$
\begin{array}{rcrr}
1 \times 0.5 & = & 0.5 & \\
1 \times 0.0 & = & 0.0 & \\
0 \times 0.8 & = & 0.0 & + \\
\hline
 & & 0.5 &
\end{array}
$$



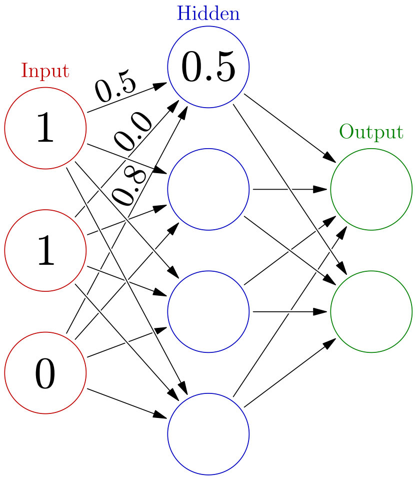
How does an artificial neural network run?

By using maths, predictably!

1. Input units are set to a *pattern*

2. Calculate hidden units' states:

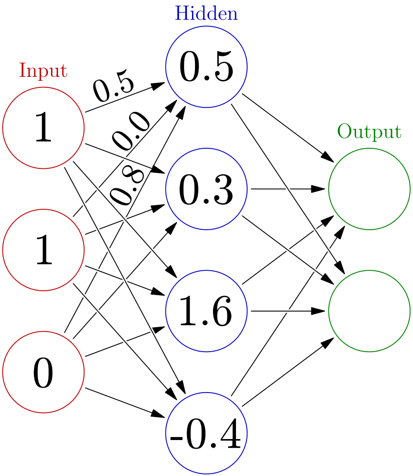
$$
\begin{array}{rcrr}
1 \times 0.5 & = & 0.5 & \\
1 \times 0.0 & = & 0.0 & \\
0 \times 0.8 & = & 0.0 & + \\
\hline
 & & 0.5 &
\end{array}
$$



How does an artificial neural network run?

By using maths, predictably!

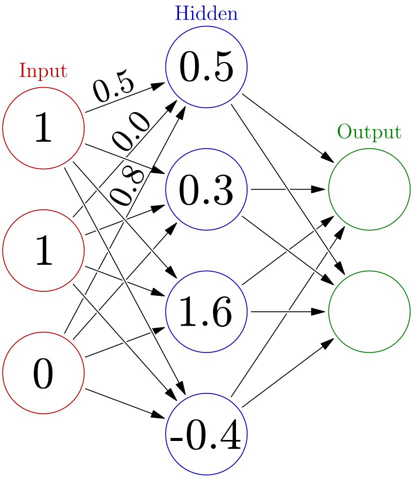
1. Input units are set to a *pattern*
2. Calculate hidden units' states



How does an artificial neural network run?

By using maths, predictably!

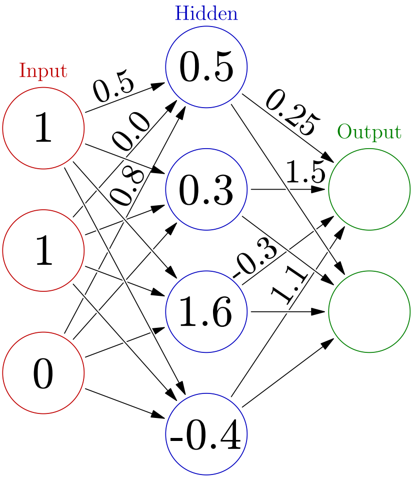
1. Input units are set to a *pattern*
2. Calculate hidden units' states
3. Same for output units



How does an artificial neural network run?

By using maths, predictably!

1. Input units are set to a *pattern*
2. Calculate hidden units' states
3. Same for output units



How does an artificial neural network run?

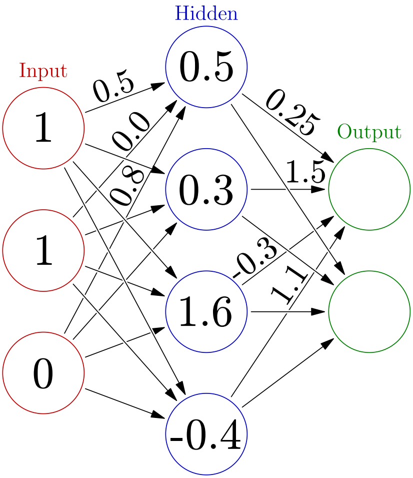
By using maths, predictably!

1. Input units are set to a *pattern*

2. Calculate hidden units' states

3. Same for output units:

$$0.5 \times 0.25 \;=\; 0.125$$

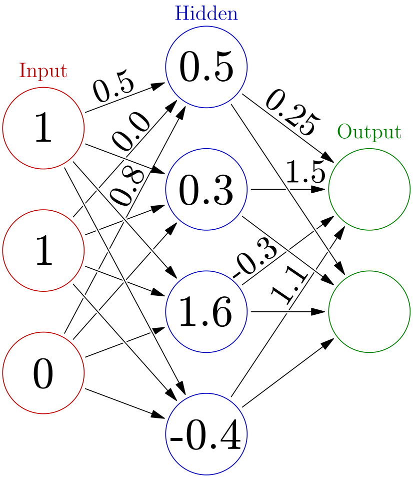


How does an artificial neural network run?

By using maths, predictably!

1. Input units are set to a *pattern*
2. Calculate hidden units' states
3. Same for output units:

$$\begin{aligned} 0.5 \times 0.25 &= 0.125 \\ 0.3 \times 1.5 &= 0.45 \end{aligned}$$



How does an artificial neural network run?

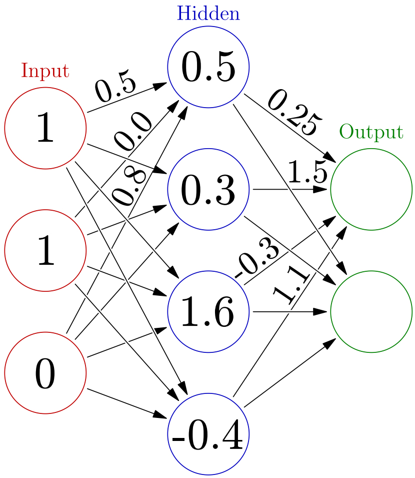
By using maths, predictably!

1. Input units are set to a *pattern*

2. Calculate hidden units' states

3. Same for output units:

$$\begin{aligned} 0.5 \times 0.25 &= 0.125 \\ 0.3 \times 1.5 &= 0.45 \\ 1.6 \times -0.3 &= -0.48 \end{aligned}$$

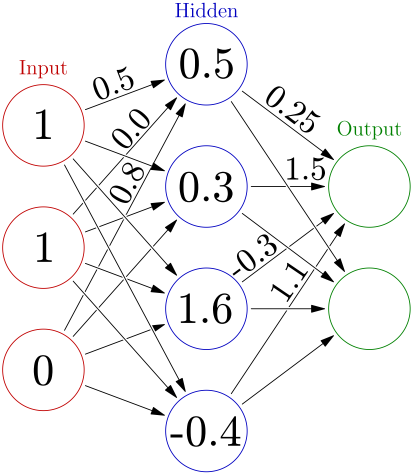


How does an artificial neural network run?

By using maths, predictably!

1. Input units are set to a *pattern*
2. Calculate hidden units' states
3. Same for output units:

$$\begin{aligned} 0.5 \times 0.25 &= 0.125 \\ 0.3 \times 1.5 &= 0.45 \\ 1.6 \times -0.3 &= -0.48 \\ -0.4 \times 1.1 &= -0.44 \end{aligned}$$



How does an artificial neural network run?

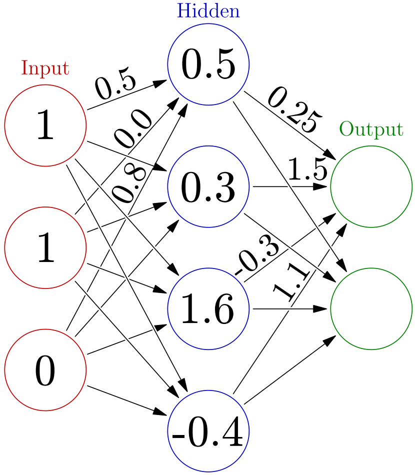
By using maths, predictably!

1. Input units are set to a *pattern*

2. Calculate hidden units' states

3. Same for output units:

$$\begin{array}{rcrl} 0.5 \times 0.25 & = & 0.125 & \\ 0.3 \times 1.5 & = & 0.45 & \\ 1.6 \times -0.3 & = & -0.48 & \\ -0.4 \times 1.1 & = & -0.44 & \\ -0.4 \times 1.1 & = & -0.44 & + \\ \hline & & -0.345 & \end{array}$$



How does an artificial neural network run?

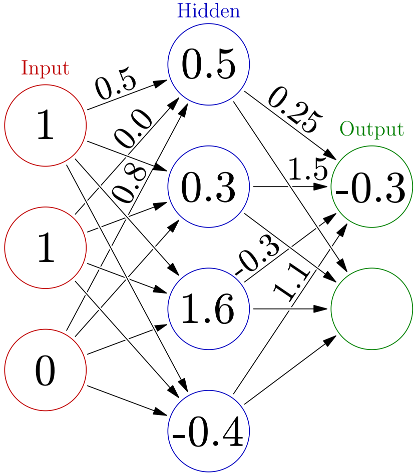
By using maths, predictably!

1. Input units are set to a *pattern*

2. Calculate hidden units' states

3. Same for output units:

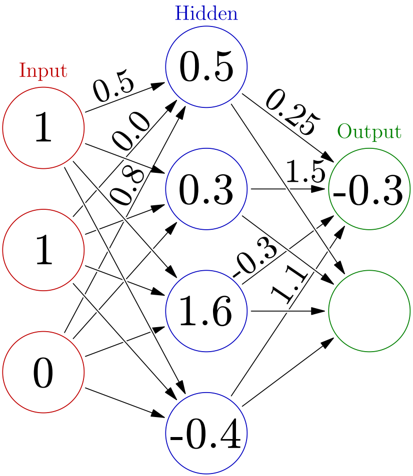
$$
\begin{array}{rcrl}
0.5 \times 0.25 & = & 0.125 & \\
0.3 \times 1.5 & = & 0.45 & \\
1.6 \times -0.3 & = & -0.48 & \\
-0.4 \times 1.1 & = & -0.44 & \\
-0.4 \times 1.1 & = & -0.44 & + \\
\hline
 & & -0.345 &
\end{array}
$$



How does an artificial neural network run?

By using maths, predictably!

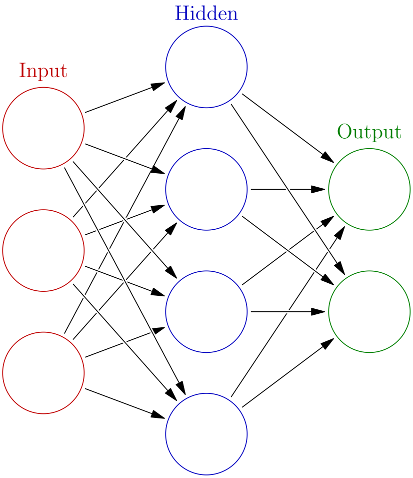
1. Input units are set to a *pattern*
2. Calculate hidden units' states
3. Same for output units



How does an artificial neural network run?

By using maths, predictably!

But we/programmers are lazy

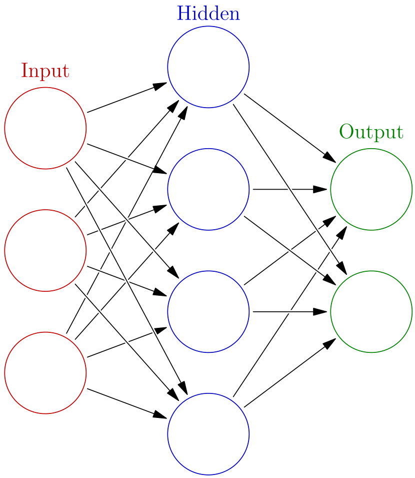


How does an artificial neural network run?

By using maths, predictably!

But we/programmers are lazy:

we like to give things general names

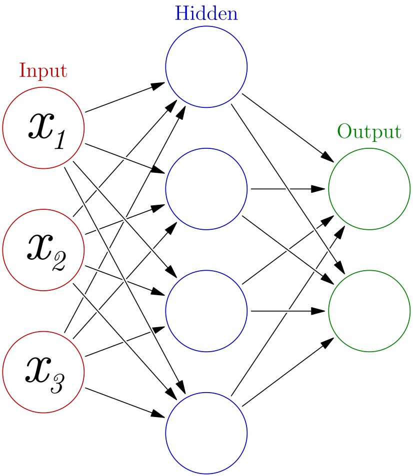


How does an artificial neural network run?

By using maths, predictably!

But we/programmers are lazy:

we like to give things general names

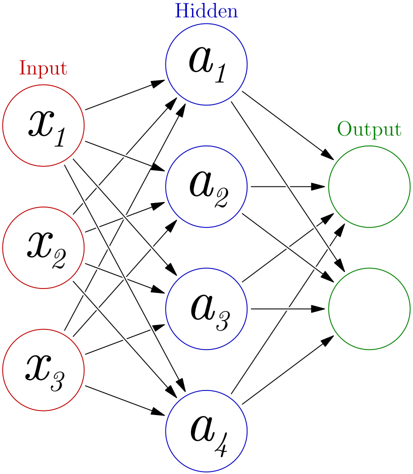


How does an artificial neural network run?

By using maths, predictably!

But we/programmers are lazy:

we like to give things general names

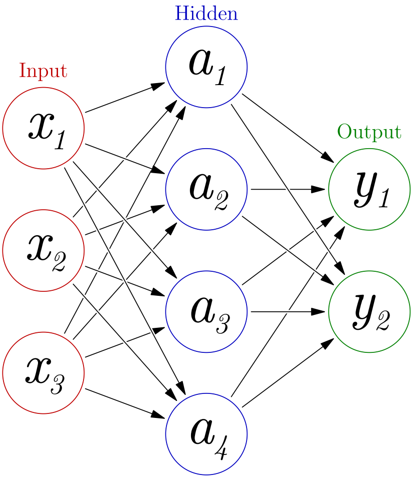


How does an artificial neural network run?

By using maths, predictably!

But we/programmers are lazy:

we like to give things general names

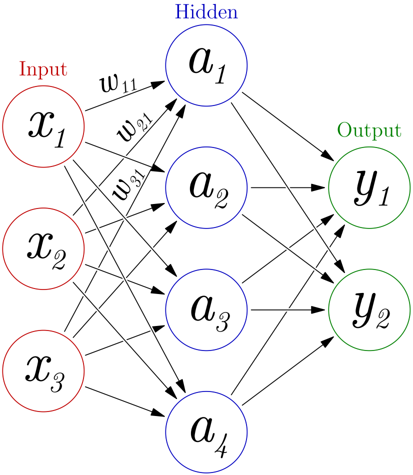


How does an artificial neural network run?

By using maths, predictably!

But we/programmers are lazy:

we like to give things general names

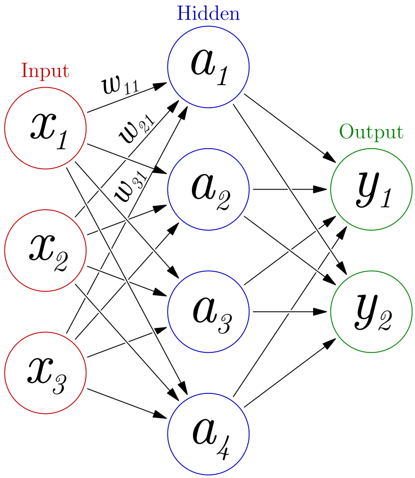


How does an artificial neural network run?

By using maths, predictably!

But we/programmers are lazy:

$a_i =$

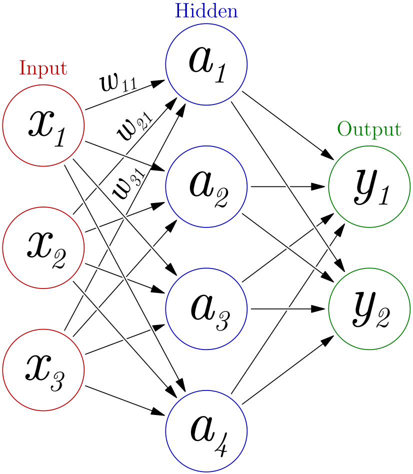


How does an artificial neural network run?

By using maths, predictably!

But we/programmers are lazy:

$$a_i = \qquad x_j \times w_{ji}$$

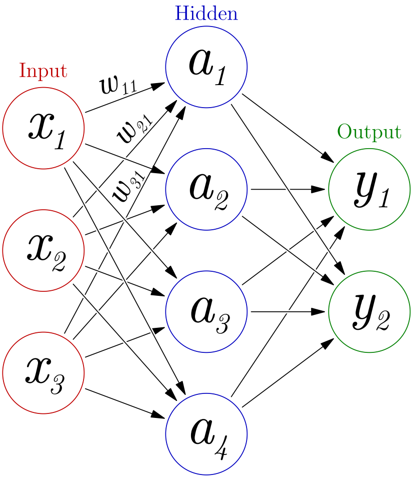


How does an artificial neural network run?

By using maths, predictably!

But we/programmers are lazy:

$$a_i = \quad \sum x_j \times w_{ji}$$

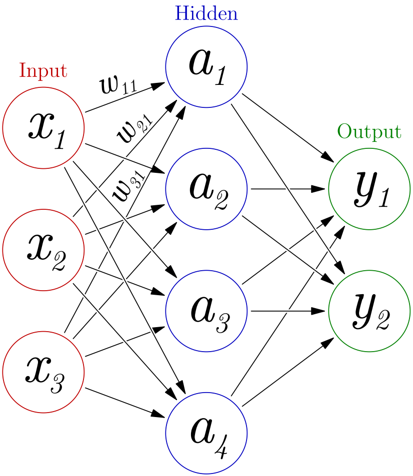


How does an artificial neural network run?

By using maths, predictably!

But we/programmers are lazy:

$$a_i = \sum_{j=1}^{N} x_j \times w_{ji}$$

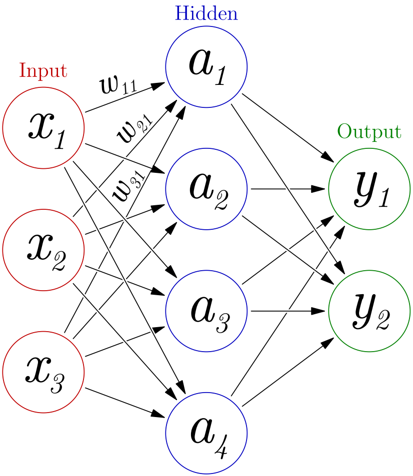


How does an artificial neural network run?

By using maths, predictably!

But we/programmers are lazy:

$$a_i = f\left(\sum_{j=1}^{N} x_j \times w_{ji}\right)$$



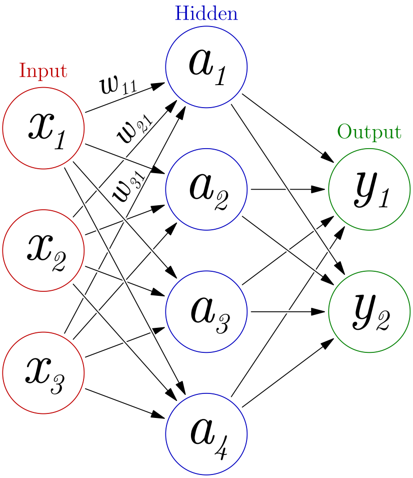
How does an artificial neural network run?

By using maths, predictably!

But we/programmers are lazy:

$$a_i = f\left(\sum_{j=1}^{N} x_j \times w_{ji}\right)$$

where a_i is the unit whose state we want to calculate, N is the number of units on the previous layer, w_{ji} is the weight on the connection between i and j, and f is a function, commonly the logistic step function.



How do networks learn?

Cunning!

- Many options: Hebbian learning, back-propagation of error, Boltzmann machine learning, self-organising map algorithm, etc.

How do networks learn?

Cunning!

- Many options: Hebbian learning, back-propagation of error, Boltzmann machine learning, self-organising map algorithm, etc.
- All learning algorithms work by changing the connection weights

How do networks learn?

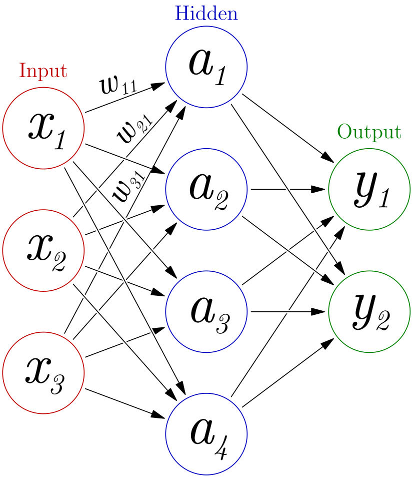
Cunning!

- Many options: Hebbian learning, back-propagation of error, Boltzmann machine learning, self-organising map algorithm, etc.
- All learning algorithms work by changing the connection weights
- Learning can be divided into *supervised*, *unsupervised*, and *reinforcement*

Hebbian learning

A very simple learning rule

> "Cells that fire together, wire together"
> — Carla Shatz

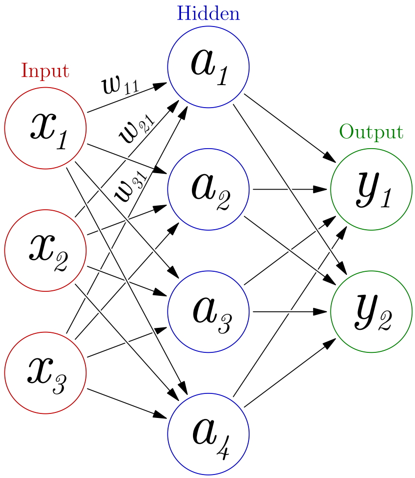


Hebbian learning

A very simple learning rule

"Cells that fire together, wire together"
— Carla Shatz

$$w_{ij} = \quad x_i \times a_j$$

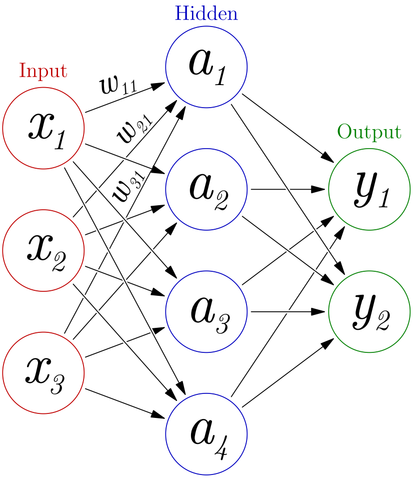


Hebbian learning

A very simple learning rule

"Cells that fire together, wire together"
— Carla Shatz

$$w_{ij} = \eta \times x_i \times a_j$$

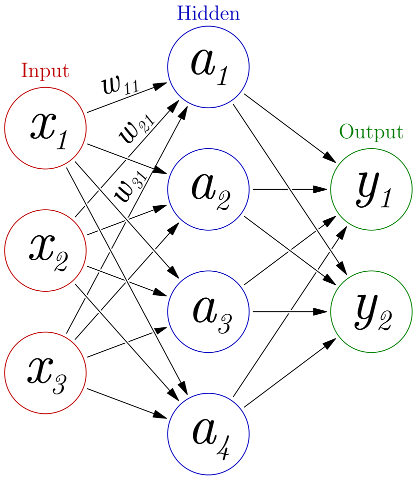


Hebbian learning

A very simple learning rule

"Cells that fire together, wire together"
— Carla Shatz

$$\Delta w_{ij} = \eta \times x_i \times a_j$$



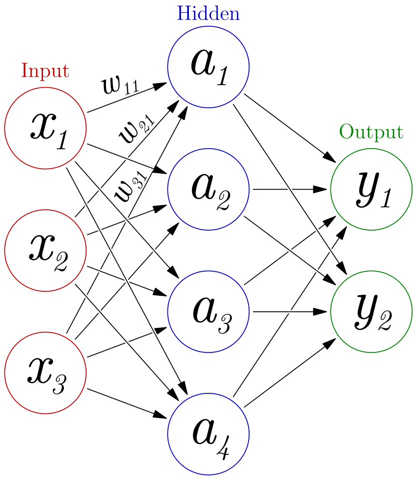
Hebbian learning

A very simple learning rule

"Cells that fire together, wire together"
— Carla Shatz

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

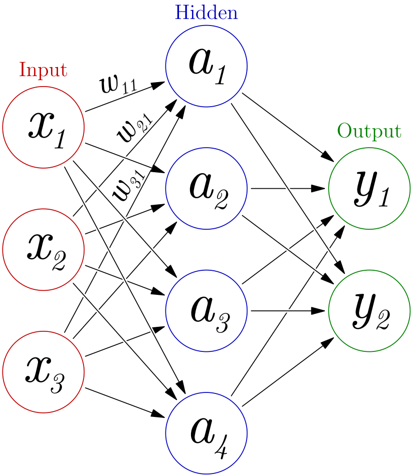
which means each weight, w_{ij} is changed by a small in/decrement for every pattern



Hebbian learning

"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

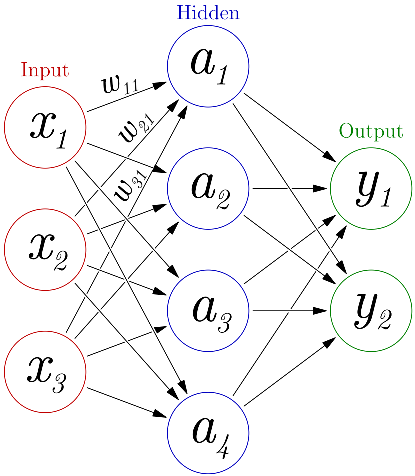


Hebbian learning

"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$



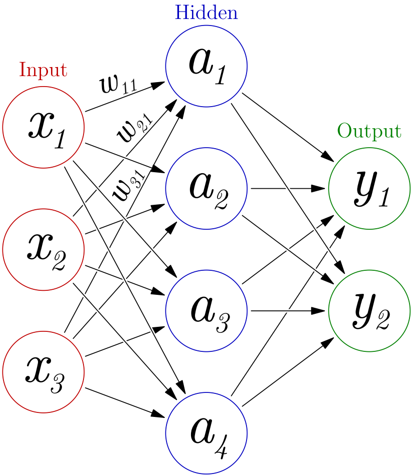
Hebbian learning

"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = \eta \times x_i \times a_j$$



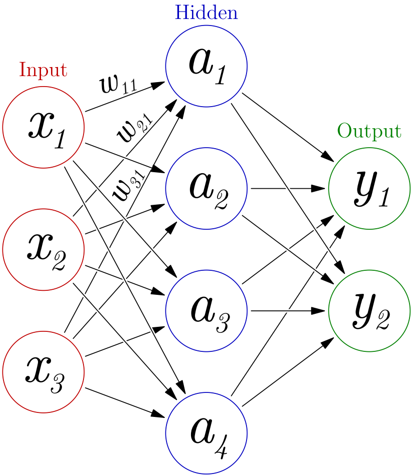
Hebbian learning

"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = 0.5 \times x_i \times a_j$$



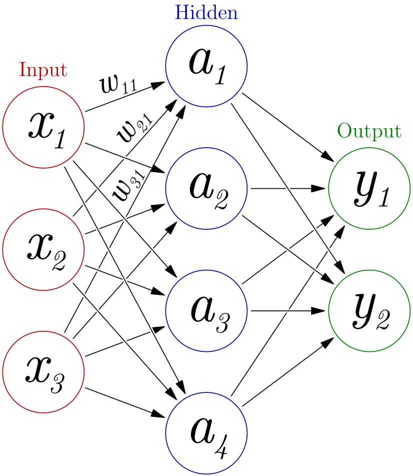
Hebbian learning

"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = 0.5 \times x_1 \times a_j$$



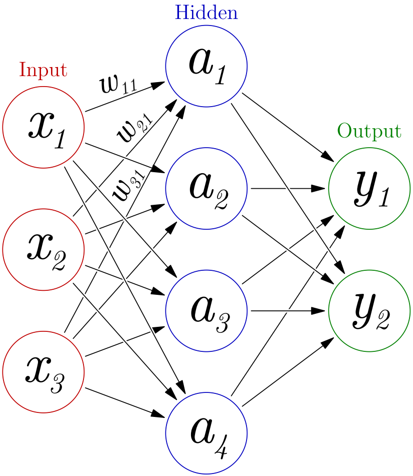
Hebbian learning

"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = 0.5 \times x_1 \times a_1$$



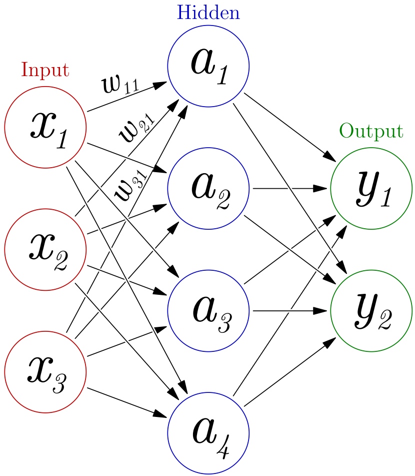
Hebbian learning

"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = 0.5 \times 1.0 \times a_1$$



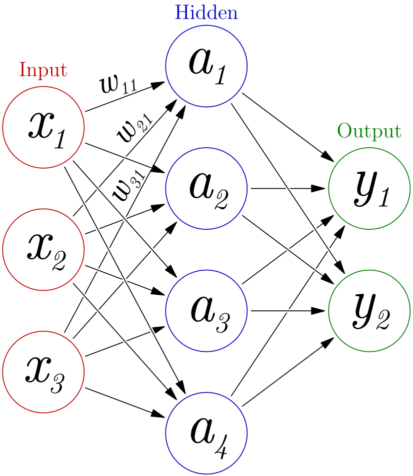
Hebbian learning

"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = 0.5 \times 1.0 \times 0.3$$



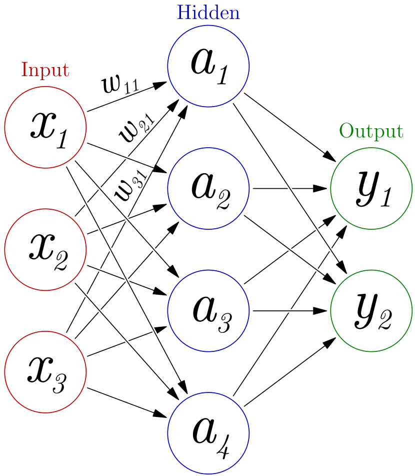
Hebbian learning

“Cells that fire together, wire together” — Carla Shatz

Hebb’s rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = 0.15$$



Hebbian learning

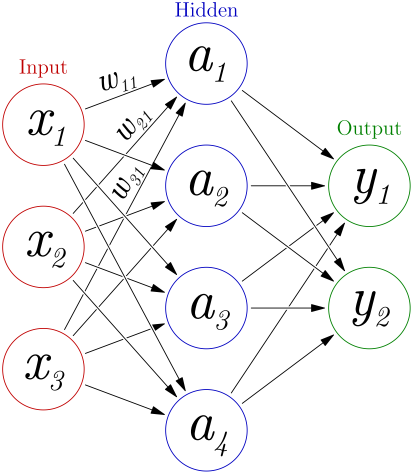
"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = 0.15$$

$$\textbf{new } w_{11} = \textbf{old } w_{11} + \Delta w_{11}$$



Hebbian learning

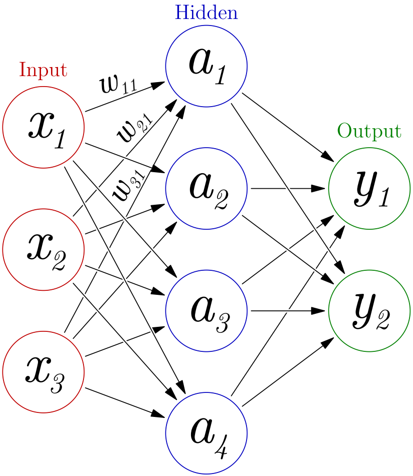
"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = 0.15$$

$$\textbf{new } w_{11} = 0.0 + \Delta w_{11}$$



Hebbian learning

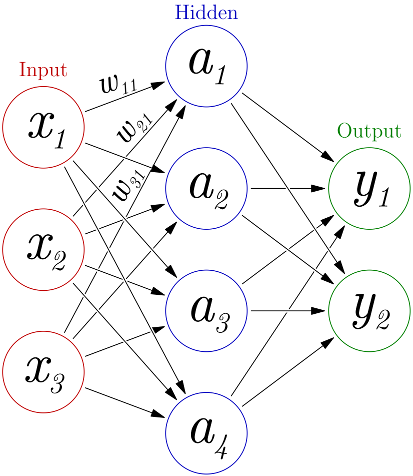
"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = 0.15$$

$$\textbf{new } w_{11} = 0.0 + 0.15$$



Hebbian learning

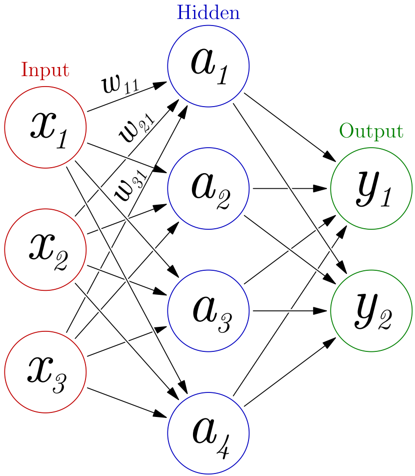
"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = 0.15$$

$$\textbf{new } w_{11} = 0.15$$



Hebbian learning

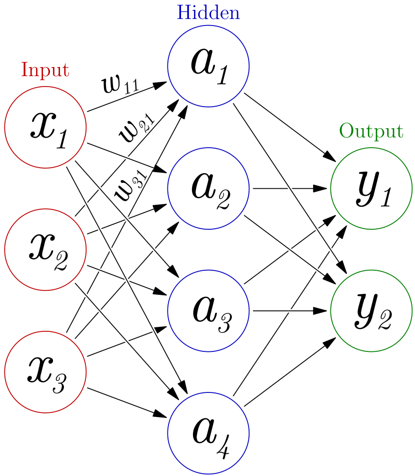
“Cells that fire together, wire together” — Carla Shatz

Hebb’s rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = 0.15$$

$$w_{11} = 0.15$$



Hebbian learning

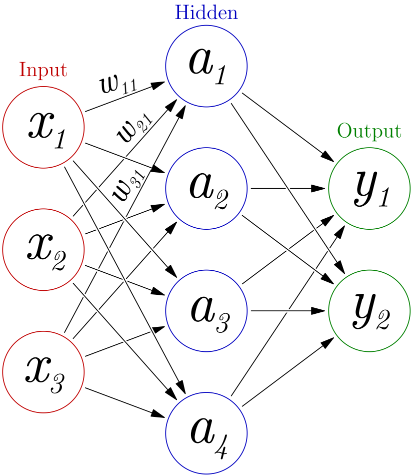
"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = 0.15$$

$$w_{11} = 0.15 + \text{something positive}$$



Hebbian learning

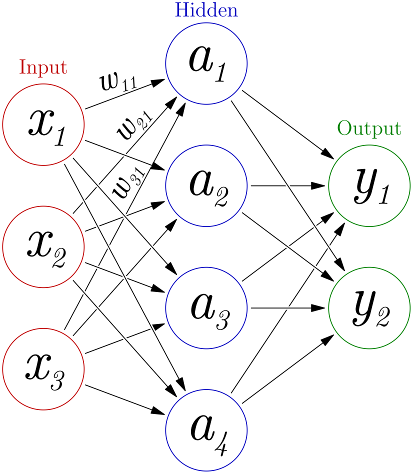
"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = 0.15$$

$$w_{11} = 0.15 + \text{something positive} + \text{something else positive} + $$



Hebbian learning

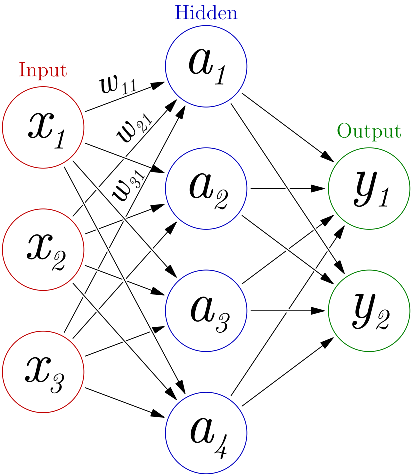
"Cells that fire together, wire together" — Carla Shatz

Hebb's rule is simple, but *very unstable*!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$

$$\Delta w_{11} = 0.15$$

$w_{11} = 0.15 +$ something positive $+$ something else positive $+$ another positive value $+ \ldots$

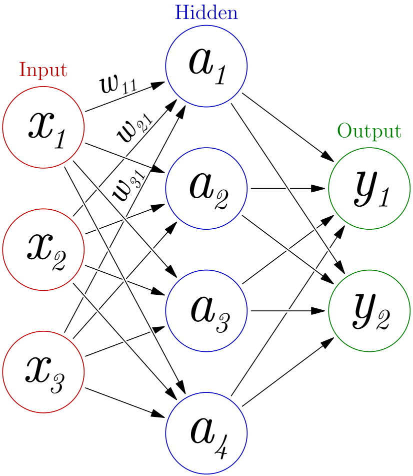


Hebbian learning

"Cells that fire together, wire together" — Carla Shatz

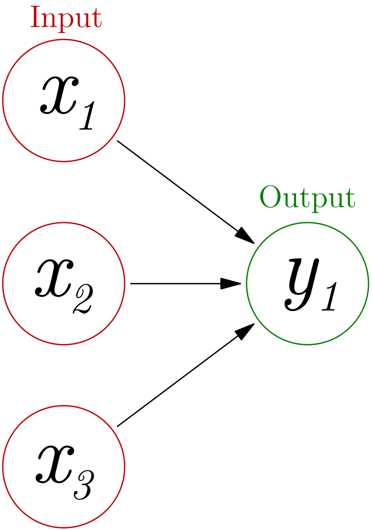
Hebb's rule is simple, but very unstable!

$$\Delta w_{ij} = \eta \times x_i \times a_j$$



The perceptron

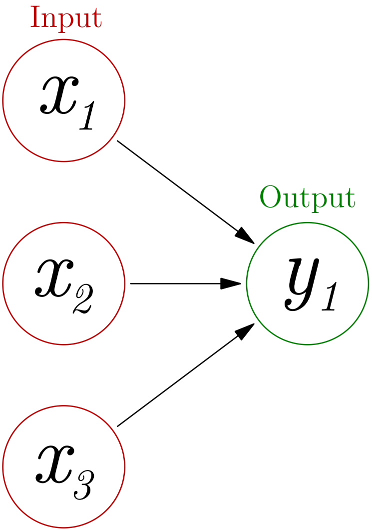
A simple classifier



The perceptron

A simple classifier

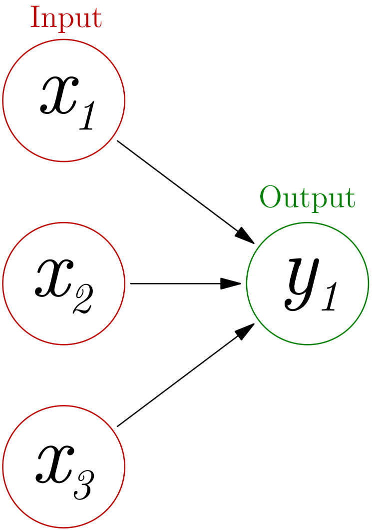
- Created in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt



The perceptron

A simple classifier

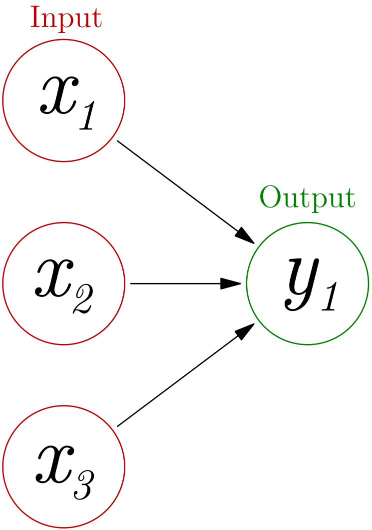
- Created in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt
- Linear classifier



The perceptron

A simple classifier

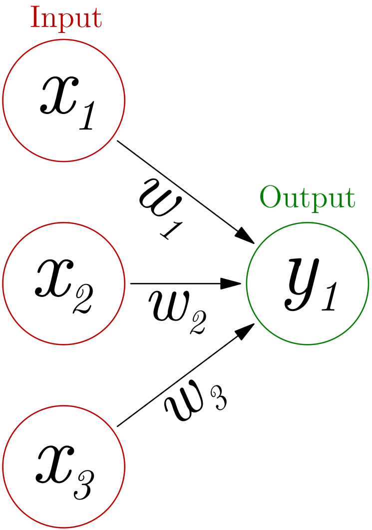
- Created in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt
- Linear classifier
- Simplest form of feedforward network



How does the perceptron learn?

Maths again!

1. Initialise weights



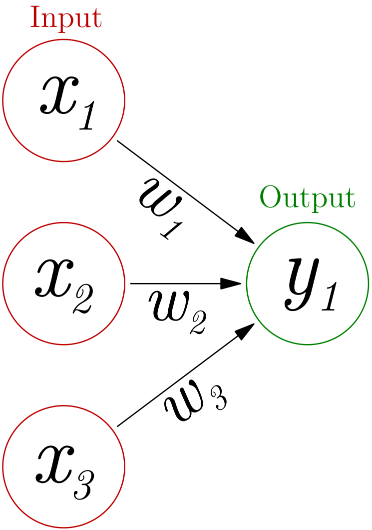
How does the perceptron learn?

Maths again!

1. Initialise weights
2. Run network using:

$$y_j = f\left(\sum_1^N w_i \times x_i\right)$$

same as always!



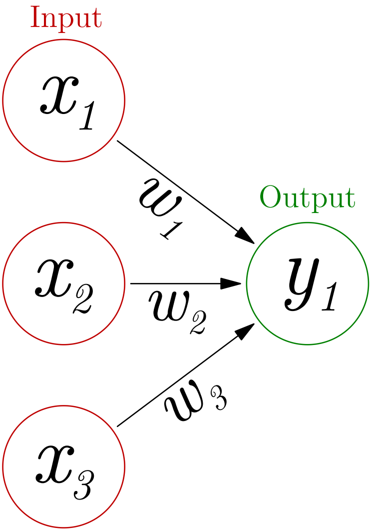
How does the perceptron learn?

Maths again!

1. Initialise weights
2. Run network
3. Update weights using:

$$\Delta w_i = \eta(d_j - y_j) \times x_i$$

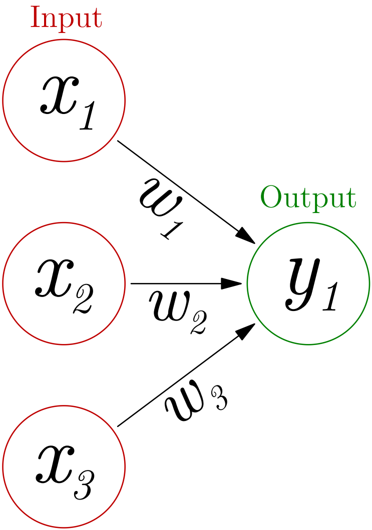
where d is what we want y to be given x_i, and η is the learning rate.



How does the perceptron learn?

Maths again!

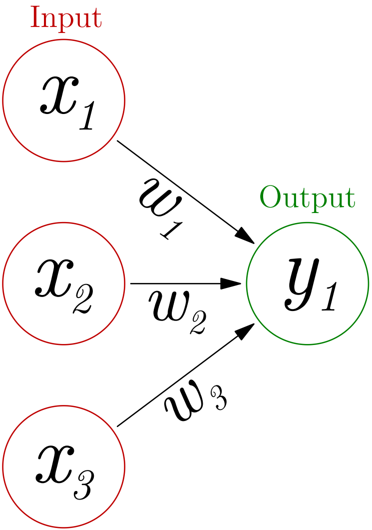
1. Initialise weights
2. Run network
3. Update weights
4. Repeat 2 and 3



How does the perceptron learn?

Maths again!

1. Initialise weights
2. Run network
3. Update weights
4. Repeat 2 and 3
5. When do we stop?



The end

Time to program a perceptron!