Bachelor of Computer Science (Hons)
Year-2 SEP2023

Welcome to Intelligent Systems

CAI3204N

Learning Objectives

- At the end of the course, students will be able to:
  - CO1: Identify the types of problem that are amenable to "intelligent" solutions.
  - CO2: Compare and contrast the various intelligent system techniques to solve such problems.
  - CO3: Select and apply appropriate intelligent techniques to a given problem.
  - CO4: Critically discuss intelligent system research issues and their applications.

Artificial Neural Networks

Revisit : More Examples

Learning Objectives

- At the end of the course, students will be able to:
  - CO1: Identify the types of problem that are amenable to "intelligent" solutions.
  - CO2: Compare and contrast the various intelligent system techniques to solve such problems.
  - CO3: Select and apply appropriate intelligent techniques to a given problem.
  - CO4: Critically discuss intelligent system research issues and their applications.

Neural Networks Problems perspective

Today's Overview

- Connectionism
- Biological Brains
- Artificial Neurons
- The Perceptron
- Learning
- Limits of Learning
- Artificial Neural Networks

http://en.wikipedia.org/wiki/Artificial_neural_network

Why ANNs

- Artificial Neural Networks are a machine learning technique inspired by the mechanism underpinning human learning
    - Lots of different types
    - Lots of different implementations
    - Lots of different applications
- If they work like humans, maybe they can learn to behave like humans?

Connectionism
Cognitive : (Complex Connection between nodes)

- An approach to Artificial Intelligence that seems to challenge the Physical Symbol System Hypothesis
- Provokes lots of interesting philosophical questions
- That we shall ignore
  - http://plato.stanford.edu/entries/connectionism

Brains

- Brains are the dominant organ of the central nervous system that controls all intelligent behaviour.
- Also much unintelligent behaviour: breathing, heart rate, blood pressure, body temperature, *etc*
- Connected to all senses and muscles throughout the body *via* spinal cord

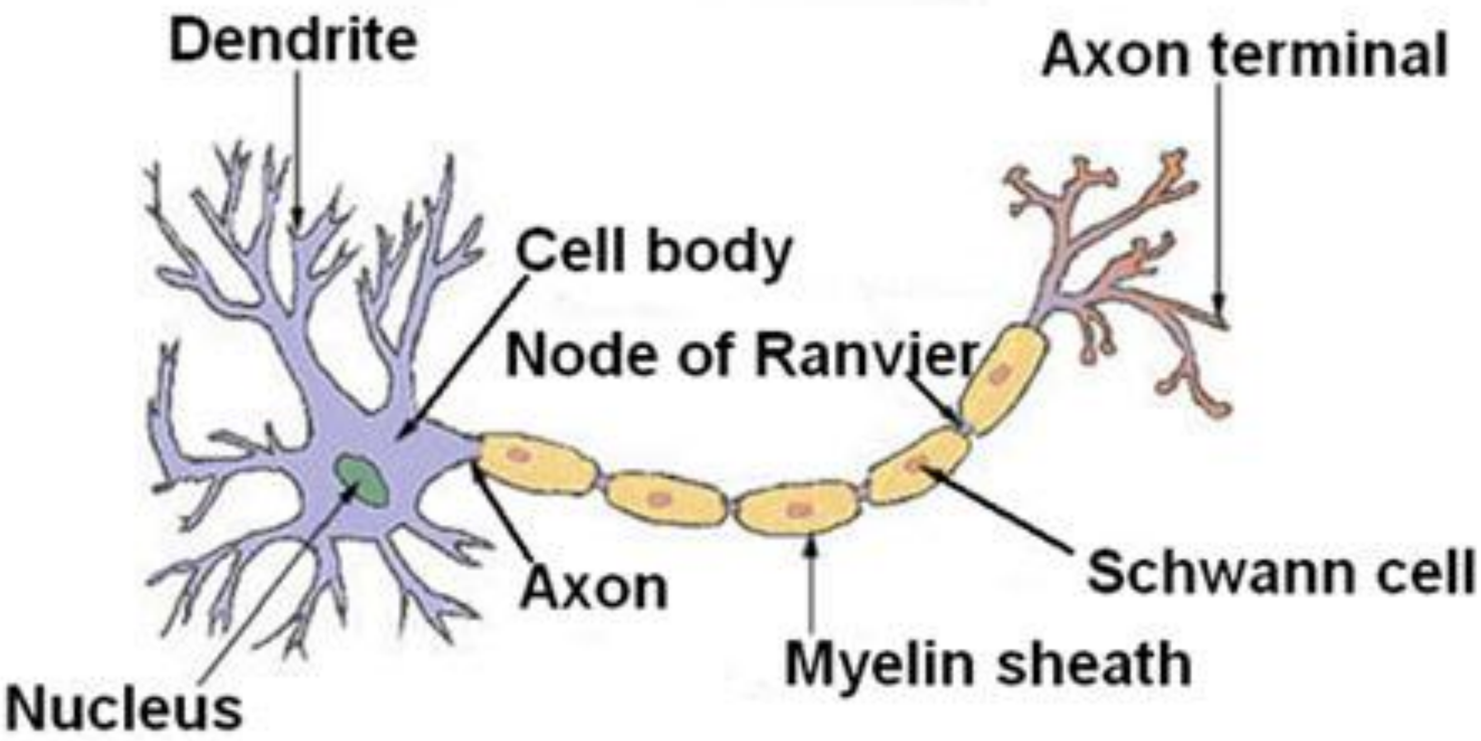
Brain Activity

- Human adult brain weighs ~3lb. *ie* about 2% of total body weight
- But consumes 25% of total energy (60% in babies!)
- Which is why you should wear a hat when its cold
- *What is going on in there?*

Neurons

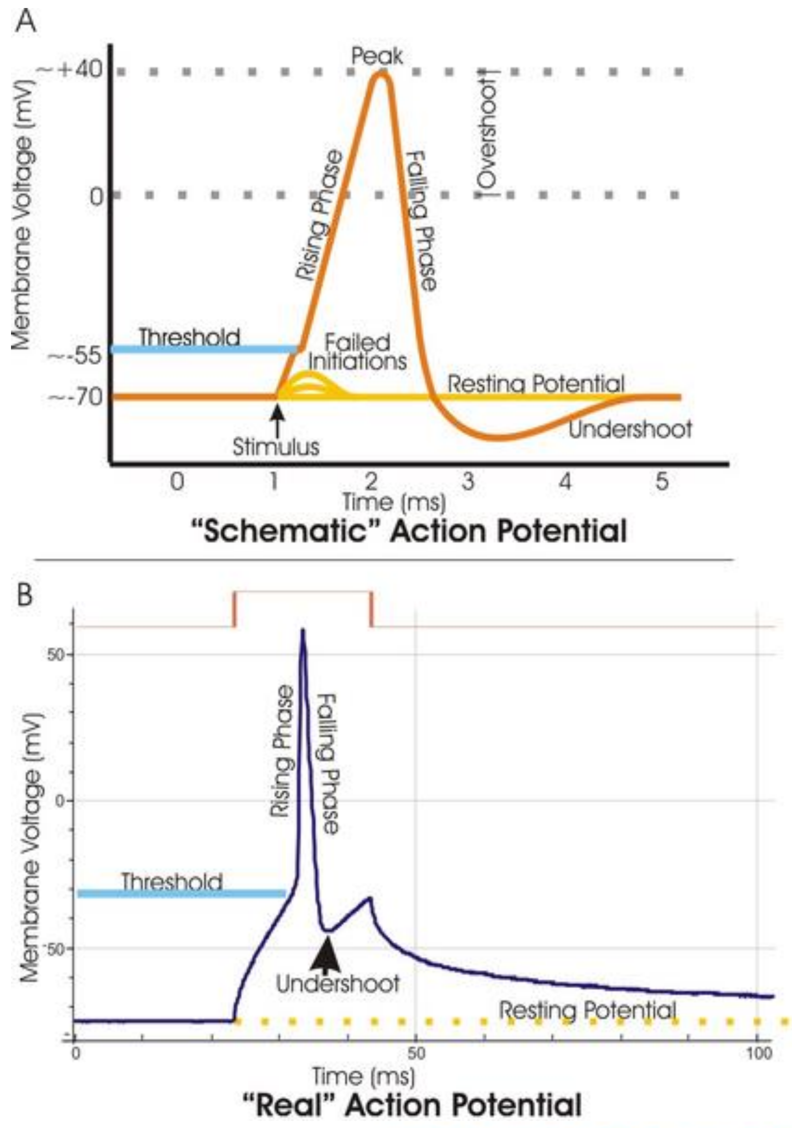
- Brains made up of neurons (and supporting glial cells)
- Neurons characterised by short and long branching connections (dendrites and axons)
- Human brain has 100 billion neurons each connected to 10,000 others

Structure of a Typical Neuron



Brain Activity

- Neurons are electrically excitable
- Gather charge from dendrites
  - (Or from sensory nerves, optic nerve, etc)
- When enough charge is gathered, neuron releases an *action potential* (spike) down the axon
  - ie when charge exceeds a threshold
- Which is transmitted on to other neurons
  - (Or on to muscle fibers)

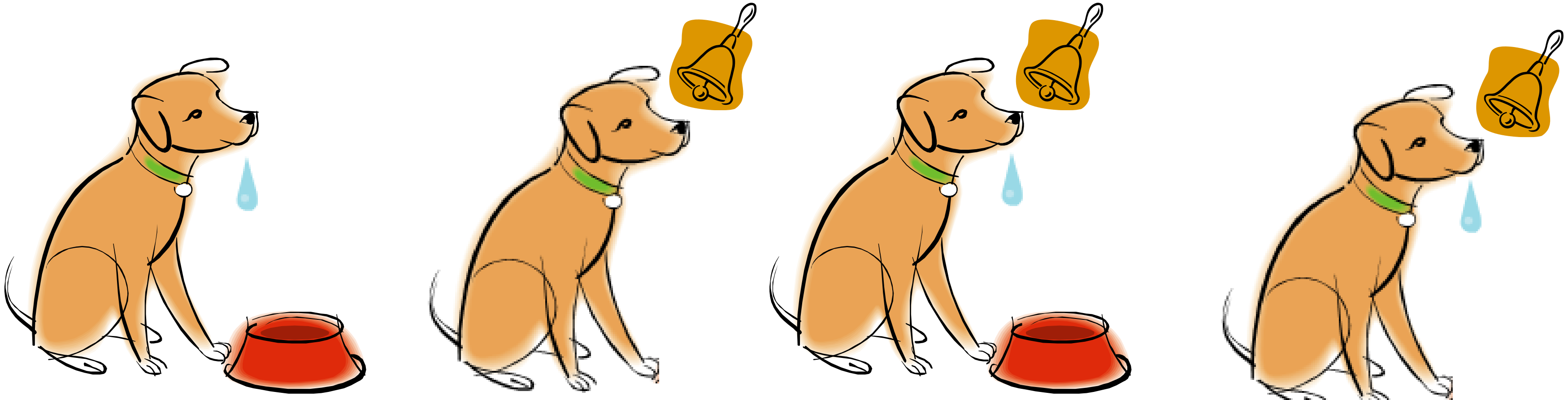


Synapses

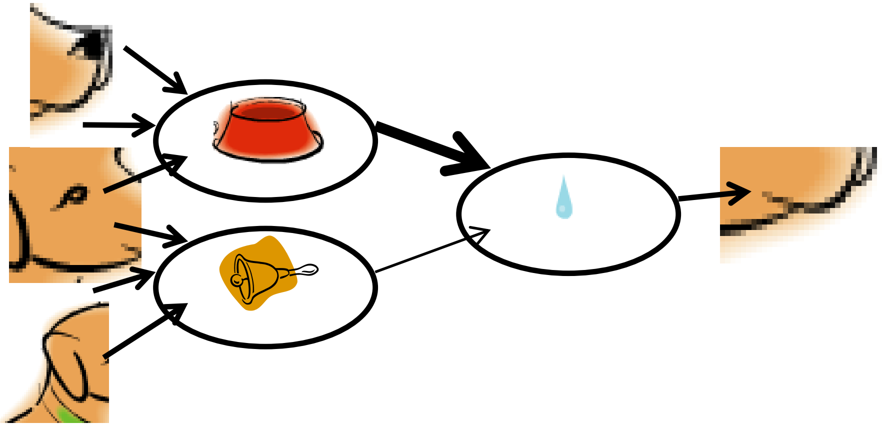
- The 'intelligence' in the brain lies in the connections
  - Which neurons are connected to which others
  - The strength of the connection (the *weight*)
- Connections can be *excitatory* or *inhibitory*

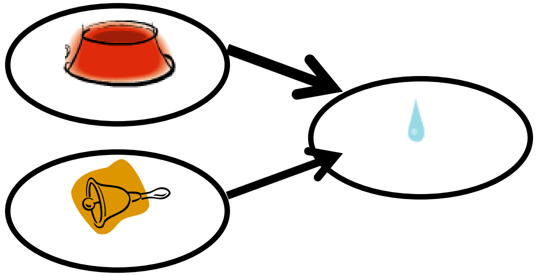
  (Serotonin Balance: stable mood)
- **Learning takes place** by changing the strength and nature of the connections.
- A lot of pschyoactive drugs work by temporarily changing the electro-chemical properties of these connections.

Hebbian Learning (Dog Training Example)



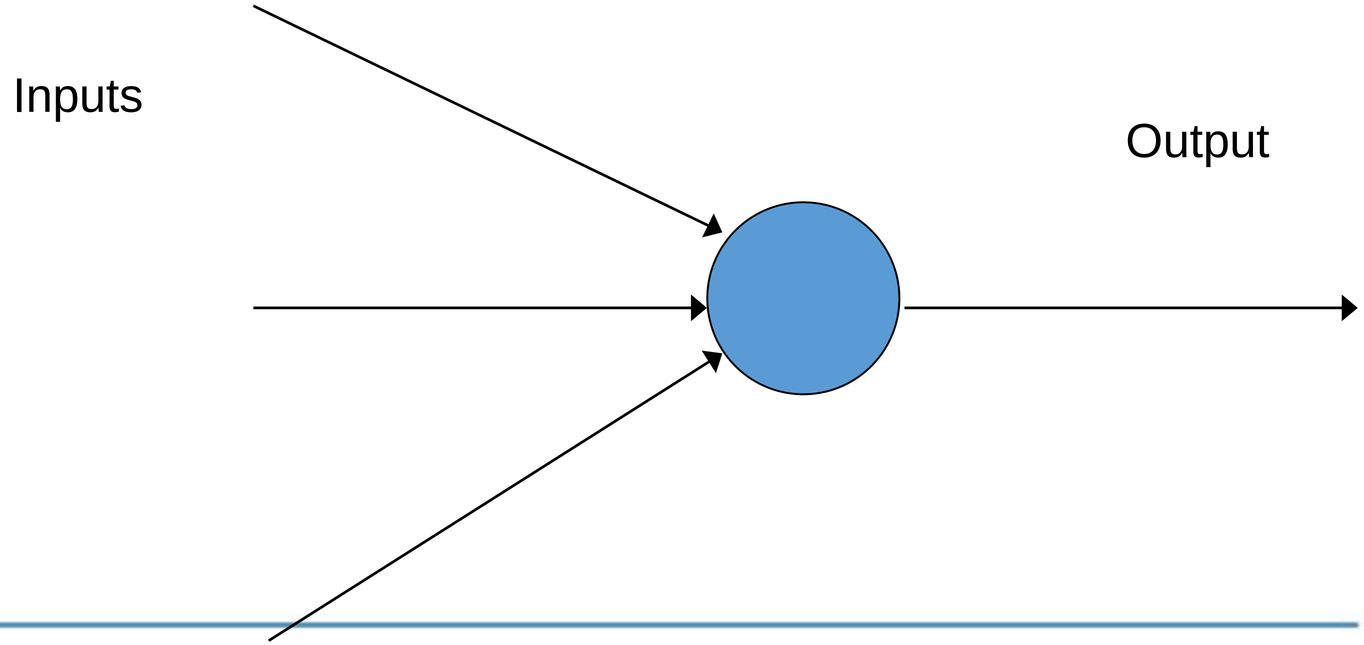
Before conditioning Conditioning After Conditioning





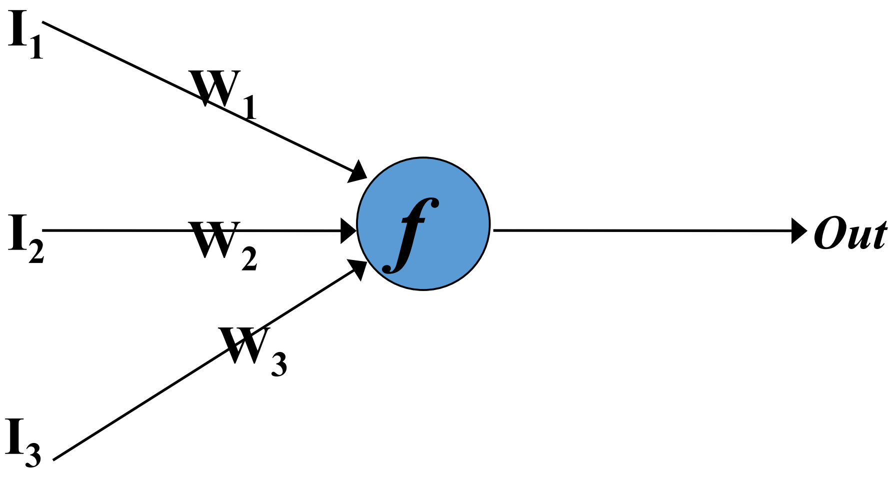
Artificial Neurons

- A cartoon model of a real neuron
- Output is a simple function of the inputs
- Exact value determined by weight of each connection, and an (optional) threshold value
- Present inputs in turn and find outputs
- Adjust weights to get behaviour we want



Activation

- Each input receives a value
- The inputs are multiplied by respective weights and added together
- Output (*Activation*) is a simple function of weighted input



$$sum = \sum I_i w_i$$
$$= I_1 w_1 + I_2 w_2 + I_3 w_3$$

$$out = f(sum)$$

Inputs to Neural Networks

- Inputs
  - Must be numeric
  - In the range [-1,1] (usually)
  - Often requires *normalisation*

Name	Values	Normalisation	Normalised Input
Rainfall	0, 5, 50mm	Divide by 100	0, 0.05, 0.5
Temperature	-5, 0, 20C	Divide by 100	-0.05, 0, .2
Stock Type	Tech, Oil, Telecoms	1 binary input for each class	(1,0,0), (0,1,0), (0,0,1)

Activation Functions

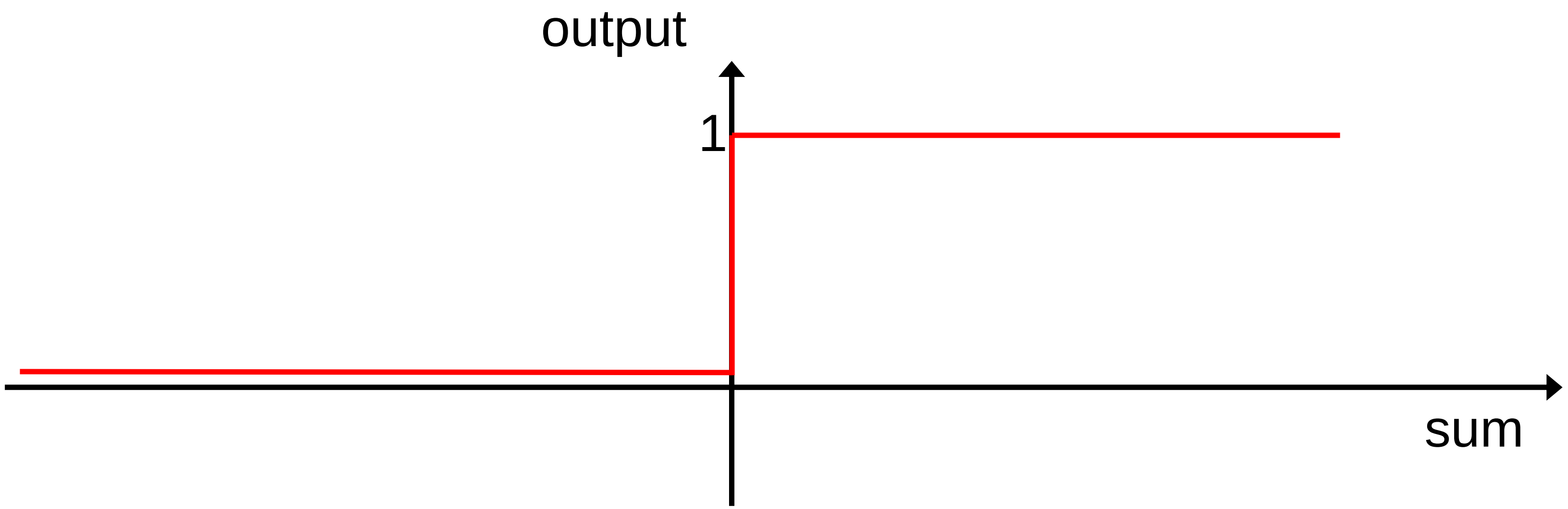
- Different types of activation function produce **different types of output.**

  Let's do this way!

- Can picture activation function as a graph of output against input sum
- The type of activation function depends on what kind of output we want
  - Binary (1 or 0, *yes* or *no*, *A* or *B*)
  - Continuous (any number)
  - Continuous range (any number 0-1)

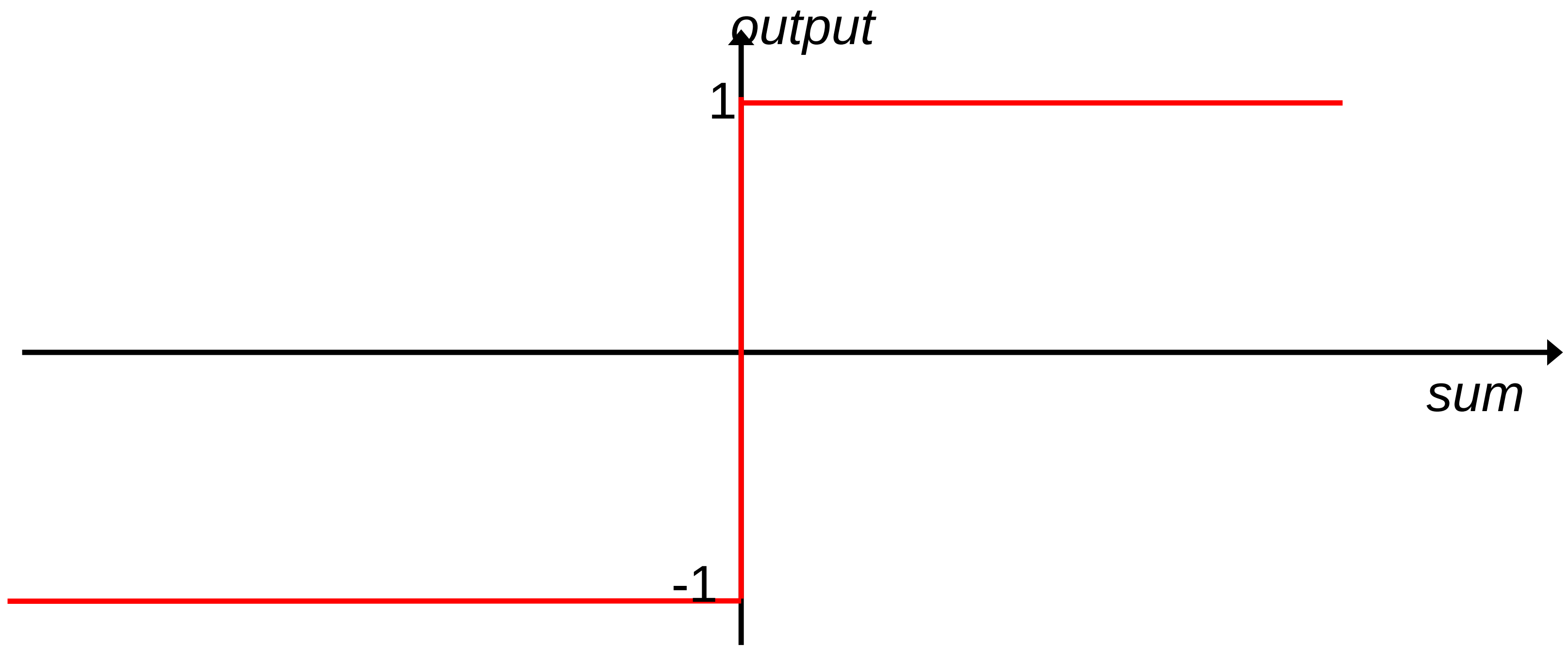
Binary Step Function

- Produces binary output (0 or 1)
- Good for yes/no type questions
- *Output = 1 if sum>0, 0 otherwise*



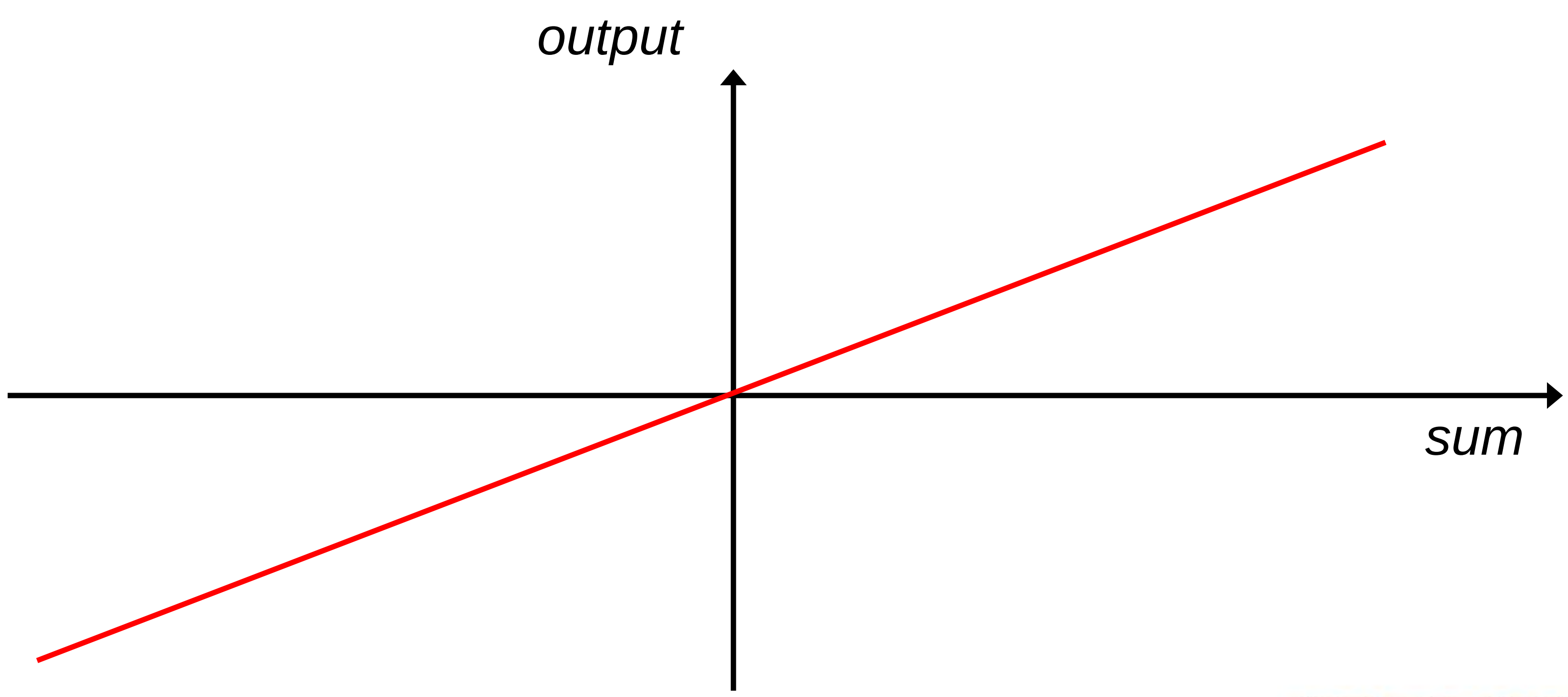
Bipolar Step Function

- Or *bipolar* output: -1 or 1
- *output = +1 if sum>0, -1 otherwise*
- (Very similar to binary, but occasionally learns better)



Identity Function

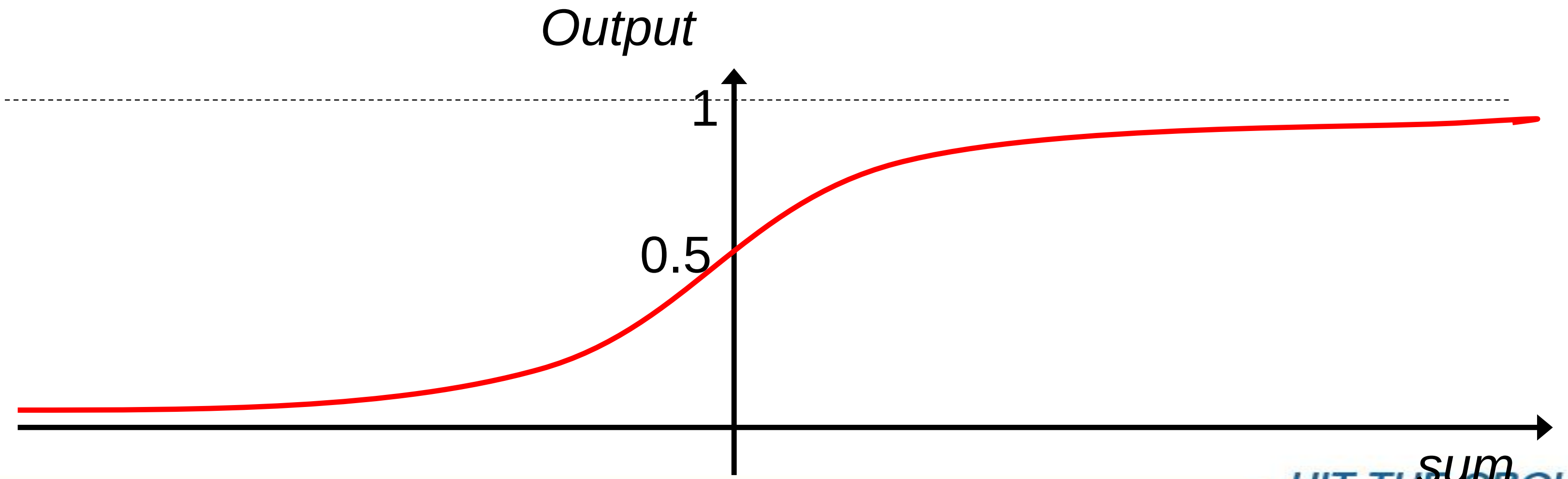
- Produces continuous output in a wide range
- $output = sum$



Sigmoidal (Squashing) Function

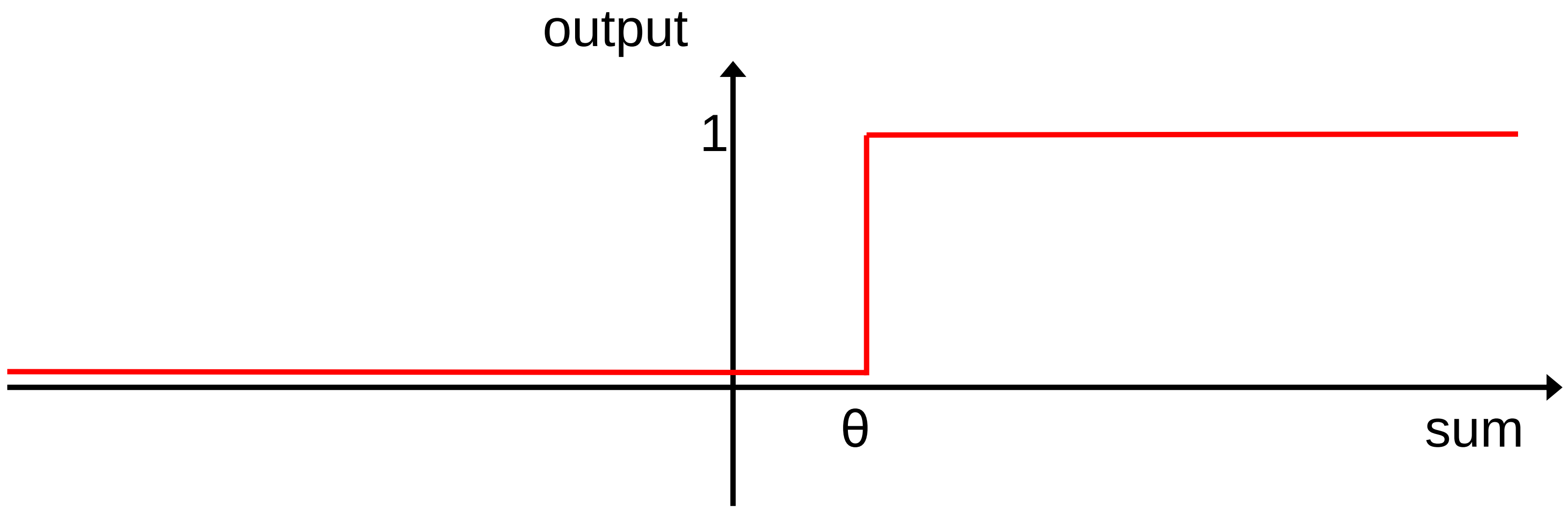
- Produces continuous output in a limited range (0,1)
- Two *asymptotes*:
  - *output→1 as sum→∞*
  - *output→0 as sum→-∞*

$$output = \frac{1}{1+e^{-sum}}$$



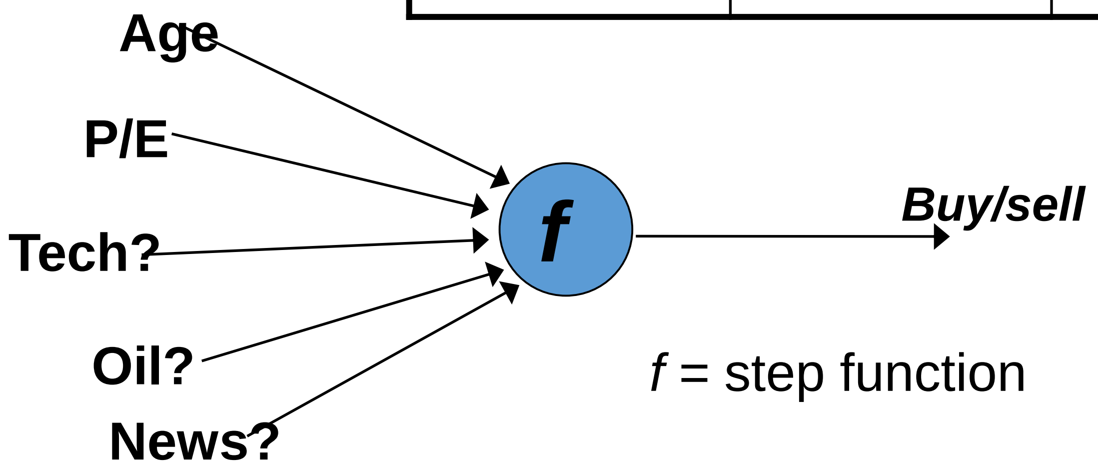
Thresholded Step Function

- Often useful to add a threshold
- *output = 1 if sum>θ, 0 otherwise,* or
- *output =1 if sum-θ>0, 0 otherwise*



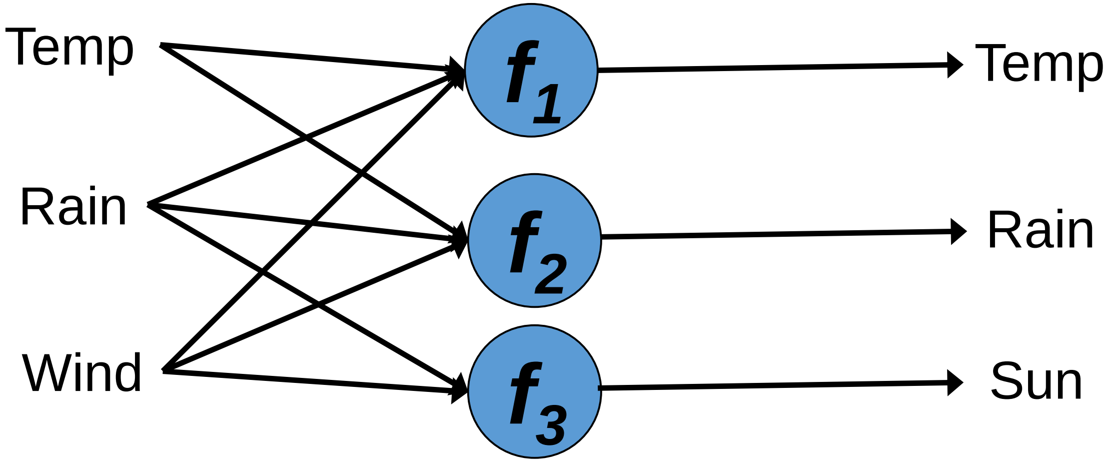
Example Networks

Company			
Age	P/E	Type	***Buy/Sell?***
12	20	Tech	*Buy*
3	15	Oil	*Sell*
5	30	News	*Buy*
20	10	News	*Sell*
10	50	Telecoms	*Sell*



f = step function

Previous Weather			Future Weather		
Temp	**Rain**	**Wind**	**Temp**	**Rain**	**Sun**
10C	5mm	20mph	12C	0mm	3hrs
12C	0mm	10mph	15C	2mm	0hrs
2C	15mm	45mph	5C	4mm	12hrs
-3C	2mm	5mph	0C	15mm	5hrs
8C	1mm	15mph	-4C	4mm	6hrs



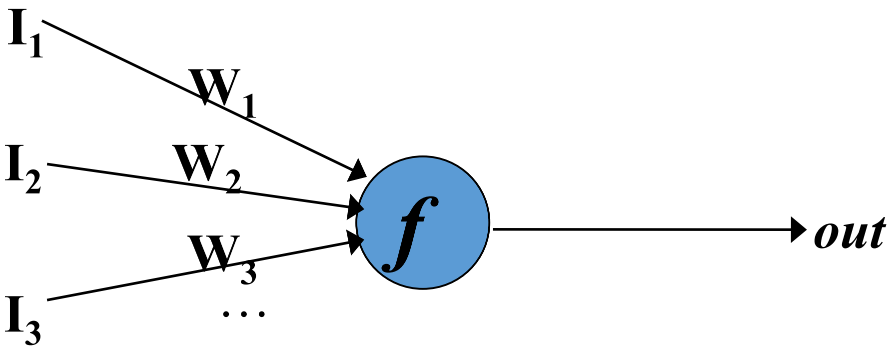
f_1, f_2 = linear function
f_3 = sigmoidal function

The Perceptron

- The simplest form of neural network
- Developed by Rosenblatt, 1957
- Used as a *decision system*
  - *ie* a two-class classifier
- Thresholded step activation function
- Binary (or sometimes bipolar) output
- http://en.wikipedia.org/wiki/Perceptron

Perceptron Thresholds

- Threshold can be represented as a weighted input

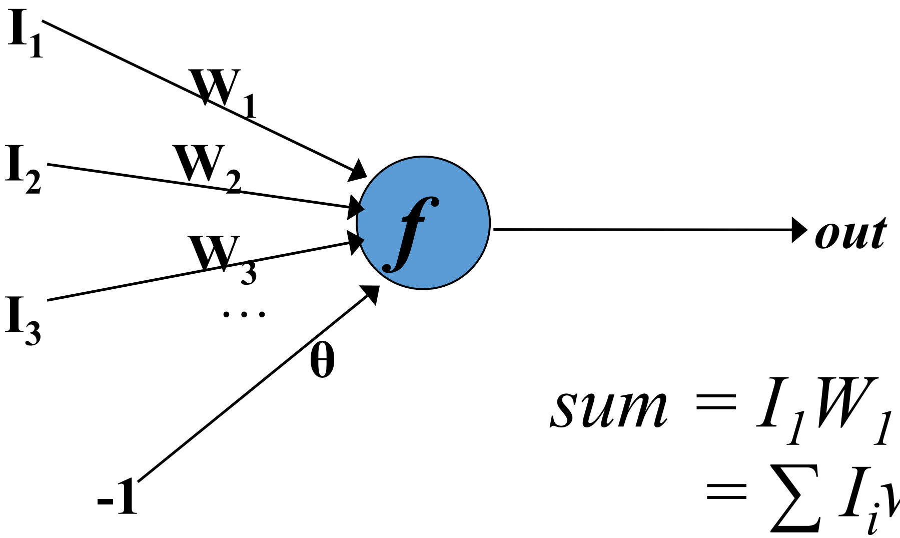


$$sum = I_1W_1 + I_2W_2 + I_3W_3 + \ldots$$
$$= \sum I_i w_i$$

$$out = 1 \text{ if } sum - \theta > 0$$
$$0 \text{ otherwise}$$

Perceptron Thresholds

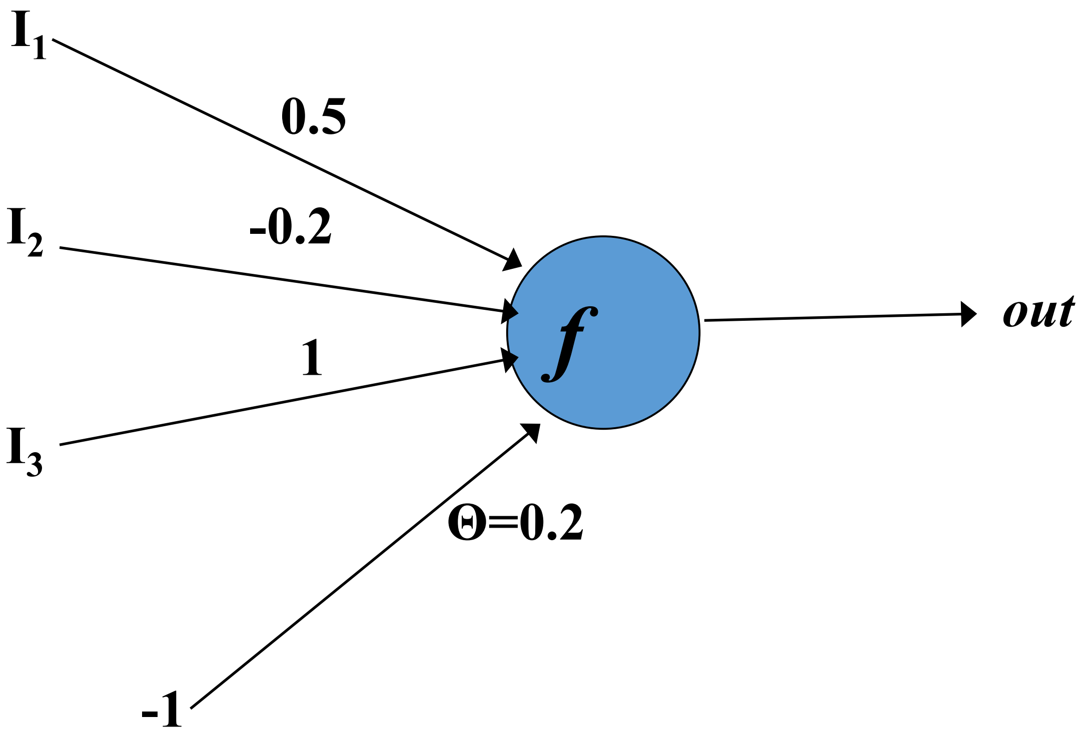
- Threshold can be represented as a weighted input



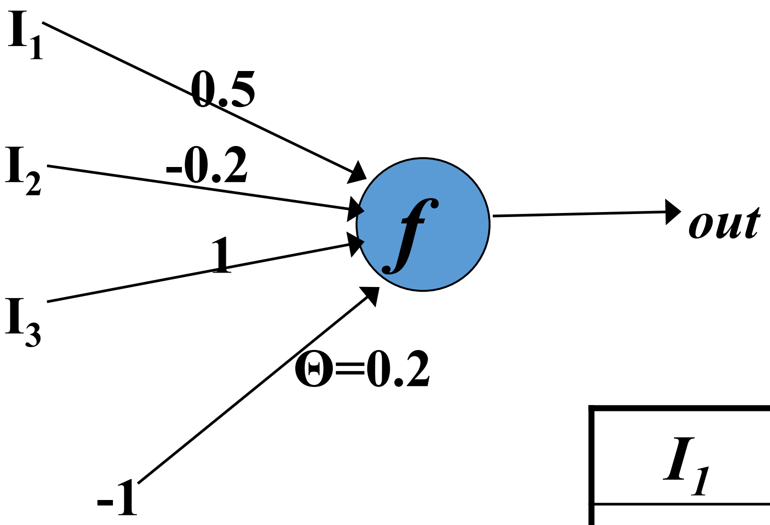
$$sum = I_1W_1 + I_2W_2 + I_3W_3 + \ldots - \theta$$
$$= \sum I_i w_i - \theta$$

$$out = 1 \text{ if } sum > 0$$
$$0 \text{ otherwise}$$

Perceptron Example



Perceptron Example



I_1	I_2	I_3	sum-θ	out
0.0	1.0	0.5	0.1	1
-0.2	0.0	-1.0	-1.3	0
0.2	-0.5	0.2	0.2	1
-0.8	0.2	-0.6	-1.44	0

Perceptron Learning

- How do we get a perceptron to learn to classify examples correctly?
- Present each of the examples in the training set
- See what output you get
- Adjust the weights to get the output ‘more right’
- Until it does what you want

Training a Perceptron

1. Start weights at random
2. Present inputs and calculate outputs
3. Find error compared with desired output
4. Adjust weights
5. Repeat 2-4 until:
    - *Either* got the outputs you want
    - *Or* results not getting any better
6. Then use network to make predictions

Learning Rule

- How to adjust the weights?
  - If input > 0:
    - If the answer is too big, reduce the weight
    - If too small, increase it
  - *but if input < 0*:
    - If the answer is too big, *increase* the weight
    - If too small, *reduce* it
  - Only increase/decrease by a little bit at a time

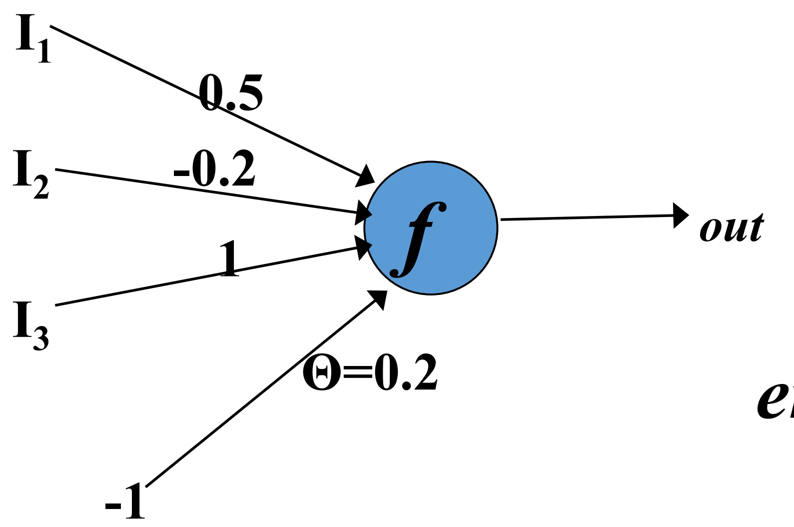
Learning Rule

$$error = output - target$$

$$rate = 0.2 \text{ (for example)}$$

$$W_{new} = W_{old} - (error \times input \times rate)$$

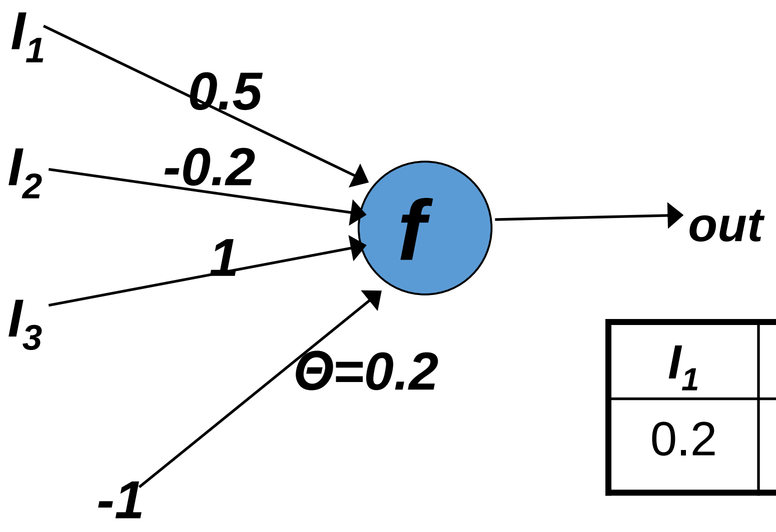
Learning Example - Error



$$error = output - target$$

I_1	I_2	I_3	$Sum\text{-}\theta$	*out*	*target*	*error*
0.0	1.0	0.5	0.1	1	1	0
0.2	-0.5	0.2	0.2	1	0	1
-0.2	0.0	-1.0	-1.3	0	1	-1
-0.8	0.2	-0.6	-1.44	0	0	0

Learning Example



$W_{new} = W_{old} - (error \times input \times rate)$

$rate = 0.2$

I_1	I_2	I_3	Sum	out	target	error
0.2	-0.5	0.2	0.2	1	0	1

error =1 ie output is too large

- I_1=0.2, so decrease W_1
- I_2=-0.5, so increase W_2
- I_3=0.2, so decrease W_3
- I_θ=-1, so increase θ

$w_{1new} = 0.5 - (1 \times 0.2 \times 0.2) = 0.46$

$w_{2new} = -0.2 - (1 \times -0.5 \times 0.2) = -0.1$

$w_{3new} = 1 - (1 \times 0.2 \times 0.2) = 0.96$

$\theta_{new} = 0.2 - (1 \times -1 \times 0.2) = 0.4$

I_1	I_2	I_3	Sum	out	target	error
0.2	-0.5	0.2	-0.066	0	0	0

Class survey(Original data)

user/film	film1	film2	film3	film4	film5	film6	film7	film8	film9	film10	film11	film12
user1	-1	1	0	-1	-1	0	0	1	0	-1	1	1
user2	1	-1	0	1	1	-1	1	-1	0	0	-1	0
user3	-1	1	0	-1	-1	1	-1	0	1	-1	1	1
user4	1	0	0	0	0	-1	1	-1	-1	1	0	-1
user5	-1	1	0	0	-1	1	0	0	0	-1	0	1
user6	0	-1	1	1	1	-1	0	-1	-1	1	-1	-1
user7	-1	0	0	0	-1	0	-1	1	1	0	0	0
user8	1	-1	1	1	1	0	0	-1	-1	1	-1	-1
user9	0	1	-1	-1	-1	0	-1	1	1	-1	1	1
user10	0	-1	1	0	1	0	0	-1	-1	0	-1	1
user11	-1	0	-1	-1	0	0	0	0	0	-1	0	0
user12	1	-1	0	1	1	0	1	-1	0	0	-1	-1

Training Data / Test Data

	user/film	film1	film2	film3	film4	film5	film6	film7	film8	film9	film10	film11	w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11	thres	sum	film12-p	film12-a	error		rate	0.1
Training Data	user1	-1	1	0	-1	-1	0	0	1	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1	1	-2			
	user4	1	0	0	0	0	-1	1	-1	-1	1	0	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-0.2	-1	-1				
	user5	-1	1	0	0	-1	1	0	0	0	-1	0	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	1				
	user6	0	-1	1	1	1	-1	0	-1	-1	1	-1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	-1				
	user8	1	-1	1	1	1	0	0	-1	-1	1	-1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	0.8	1	-1				
	user9	0	1	-1	-1	-1	0	-1	1	1	-1	1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	1				
	user1	-1	1	0	-1	-1	0	0	1	0	-1	1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	1				
	user4	1	0	0	0	0	-1	1	-1	-1	1	0	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-0.2	-1	-1				
	user5	-1	1	0	0	-1	1	0	0	0	-1	0	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	1				
	user6	0	-1	1	1	1	-1	0	-1	-1	1	-1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	-1				
	user8	1	-1	1	1	1	0	0	-1	-1	1	-1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	0.8	1	-1				
	user9	0	1	-1	-1	-1	0	-1	1	1	-1	1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	1				
	user1	-1	1	0	-1	-1	0	0	1	0	-1	1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	1				
	user4	1	0	0	0	0	-1	1	-1	-1	1	0	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-0.2	-1	-1				
	user5	-1	1	0	0	-1	1	0	0	0	-1	0	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	1				
	user6	0	-1	1	1	1	-1	0	-1	-1	1	-1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	-1				
	user8	1	-1	1	1	1	0	0	-1	-1	1	-1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	0.8	1	-1				
	user9	0	1	-1	-1	-1	0	-1	1	1	-1	1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	1				
	user1	-1	1	0	-1	-1	0	0	1	0	-1	1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	1				
	user4	1	0	0	0	0	-1	1	-1	-1	1	0	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-0.2	-1	-1				
	user5	-1	1	0	0	-1	1	0	0	0	-1	0	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	1				
	user6	0	-1	1	1	1	-1	0	-1	-1	1	-1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	-1				
	user8	1	-1	1	1	1	0	0	-1	-1	1	-1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	0.8	1	-1				
	user9	0	1	-1	-1	-1	0	-1	1	1	-1	1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	1				
	user1	-1	1	0	-1	-1	0	0	1	0	-1	1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	1				
	user4	1	0	0	0	0	-1	1	-1	-1	1	0	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-0.2	-1	-1				
	user5	-1	1	0	0	-1	1	0	0	0	-1	0	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	1				
	user6	0	-1	1	1	1	-1	0	-1	-1	1	-1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	-1				
	user8	1	-1	1	1	1	0	0	-1	-1	1	-1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	0.8	1	-1				
	user9	0	1	-1	-1	-1	0	-1	1	1	-1	1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-1.2	-1	1				
Test Data	user10	0	-1	1	0	1	0	0	-1	-1	0	-1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-2.2	-1	1				
	user11	-1	0	-1	-1	0	0	0	0	0	-1	0	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	-4.2	-1	1				
	user12	1	-1	0	1	1	0	1	-1	0	0	-1	-0.2	0.2	0	-0.2	-0	0	0	0.2	0	-0	0.2	0	0.8	-1	1				

Sheet1

Training Regimes

- Suppose there are three examples in training set
  - {A,B,C}
  - Present each example repeatedly until successful on each
  - *Or* Present complete set in turn until successful on all?
- Which training regime would be best?
- The latter, since by the time we have trained on B and C, may have 'forgotten' how to do A

A ×
A ×
A ×
A √
B ×
B ×
B √
C ×
C ×
C ×
C √

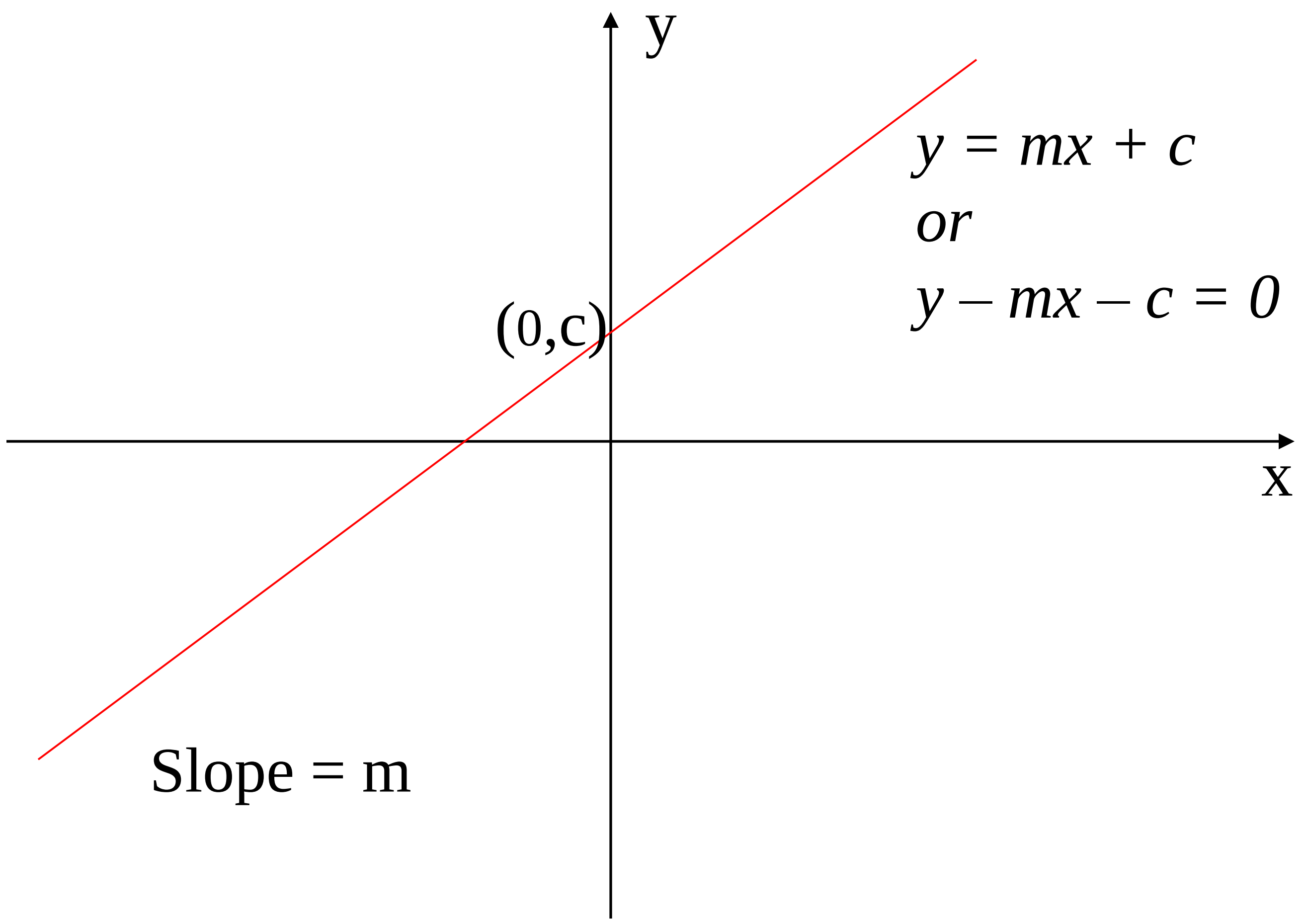
OR

A ×
B ×
C √
A √
B ×
C √
A ×
B ×
C √
…
A √
B √
C √

Limits of the Perceptron

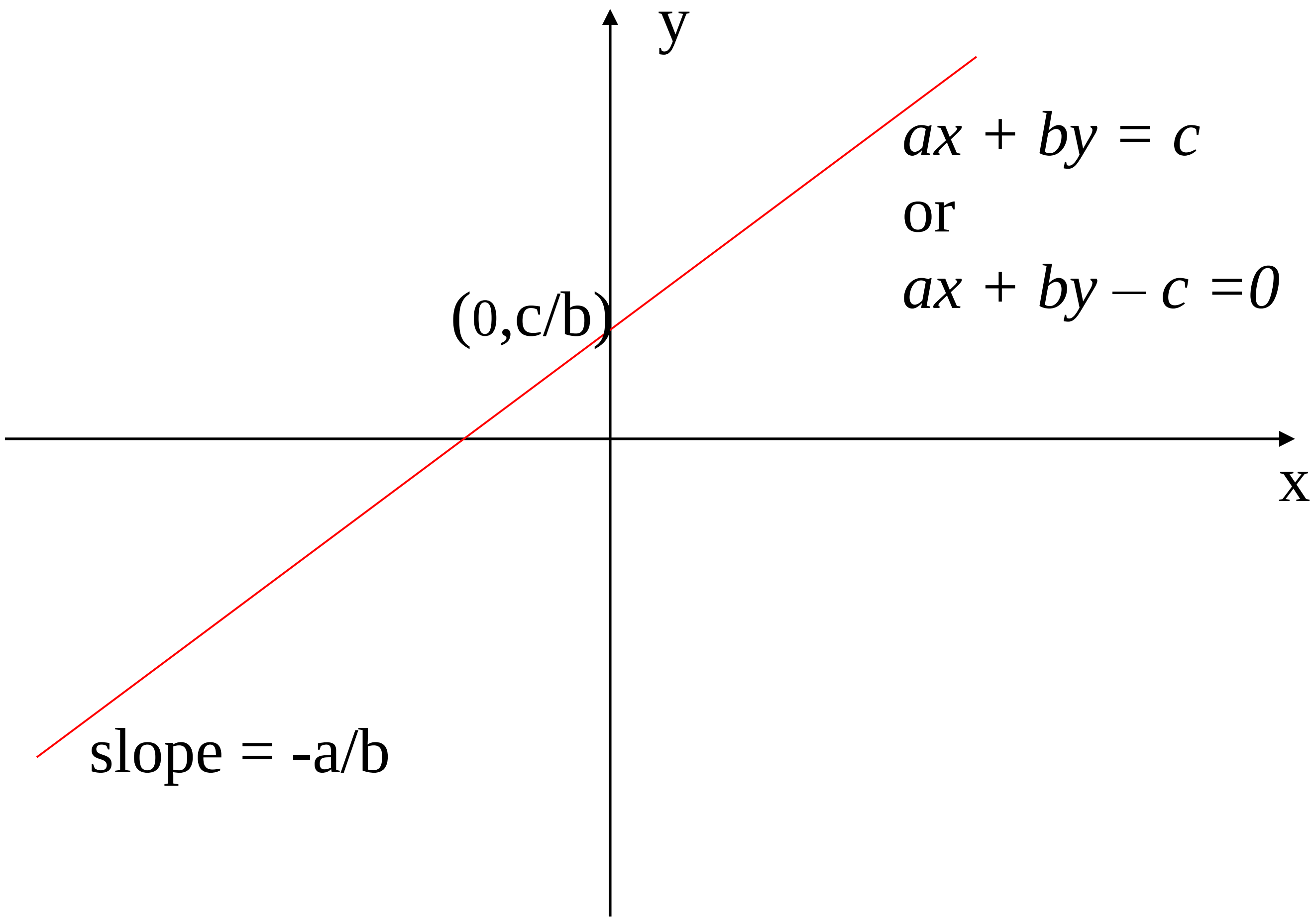
- Perceptron was invented in 50s
- Seemed to promise much
- But in 1969 Minsky and Papert proved there were limits on what it could learn…

Straight Line in the Plane

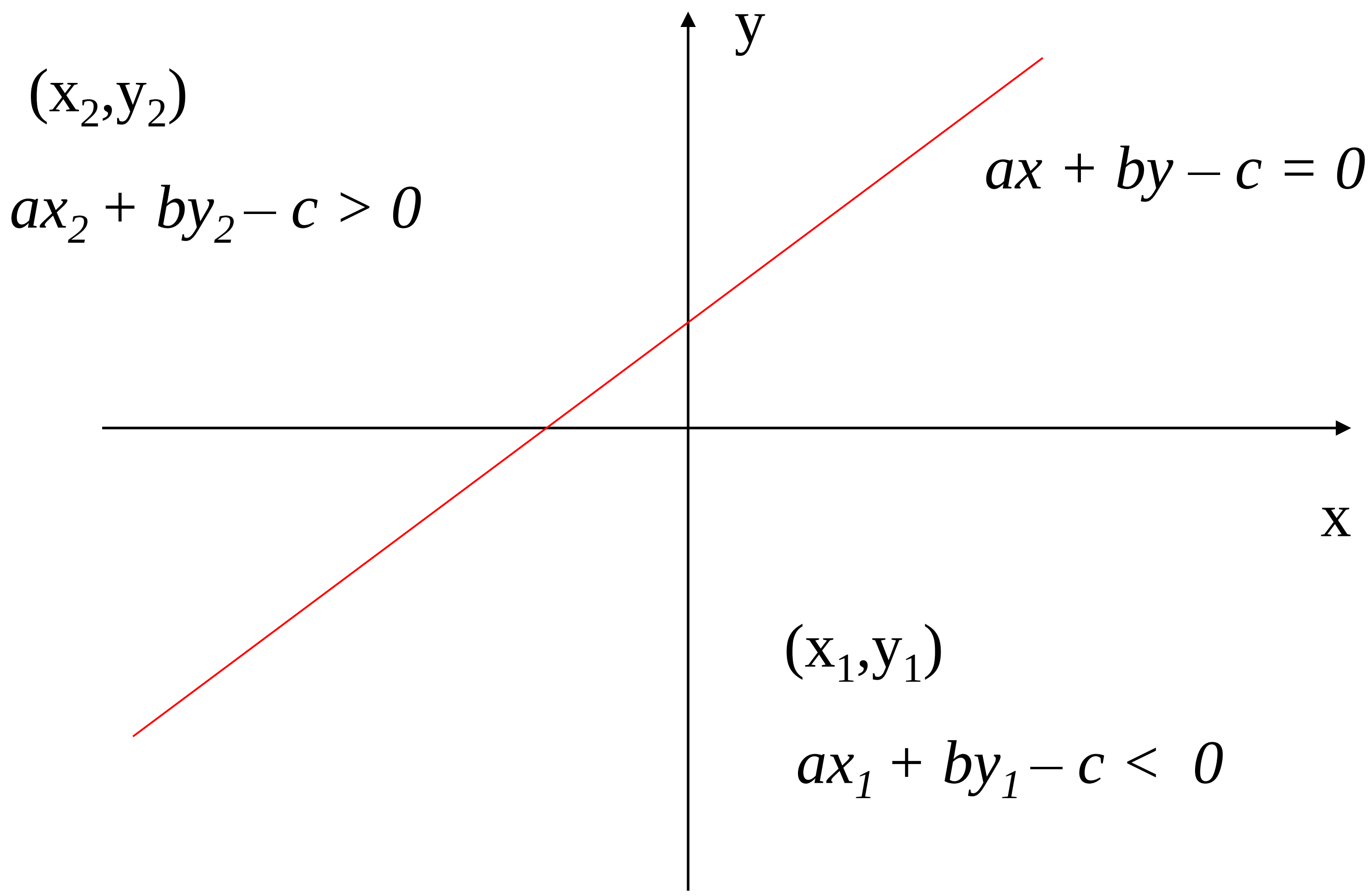


But this form of the equation cannot describe a vertical line!

Or, more generally

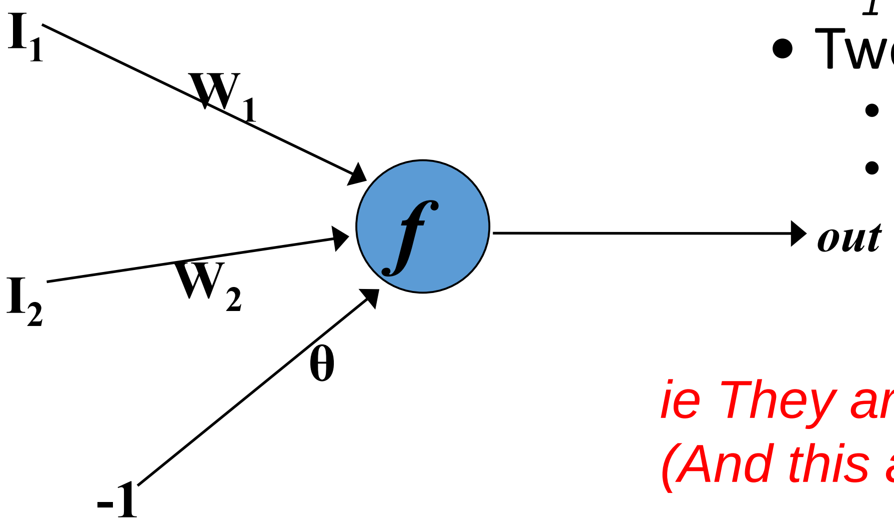


Linear Separation



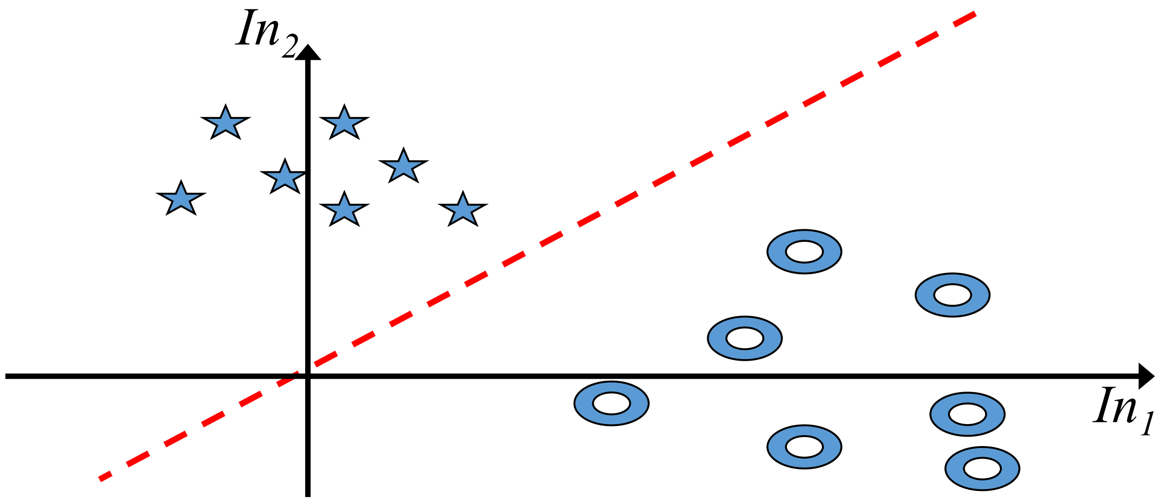
Linear Perceptron

- Equation for the line:
  - $ax + by - c = 0$
  - Two regions
    - $ax + by - c > 0$
    - $ax + by - c < 0$
- Equation for the perceptron:
  - $w_1 in_1 + w_2 in_2 - \vartheta = sum$
  - Two possible outputs
    - $w_1 in_1 + w_2 in_2 - \vartheta > 0$
    - $w_1 in_1 + w_2 in_2 - \vartheta < 0$



ie They are of the same form!
(And this applies to >2 inputs/dimensions)

Linear Perceptron

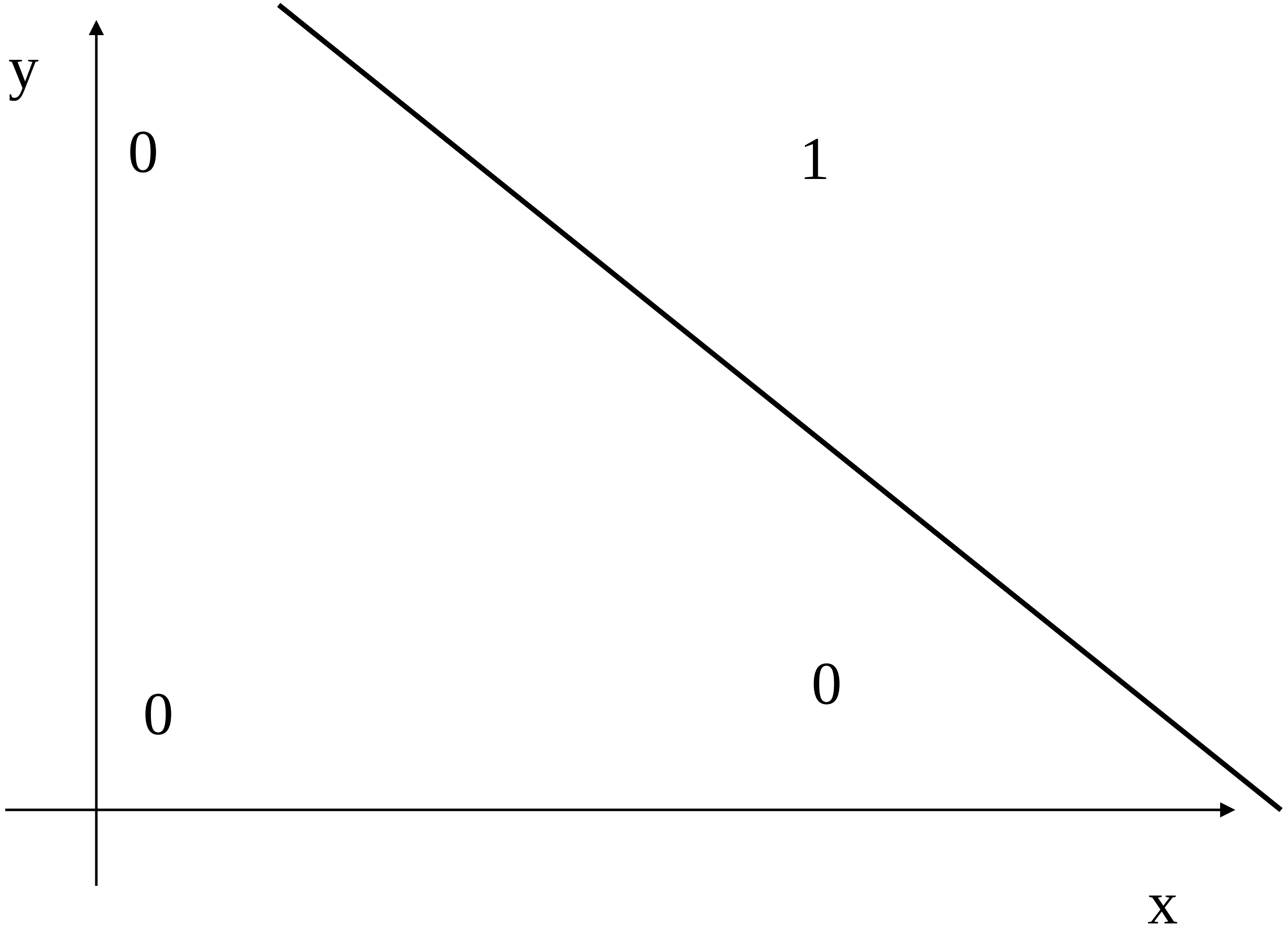


- A single perceptron can learn to solve a problem if it is *linearly seperable*
- *ie* if the two classes can be seperated by a single straight line (or hyperplane)

Example Problems

- Logical AND
- Linearly separable?

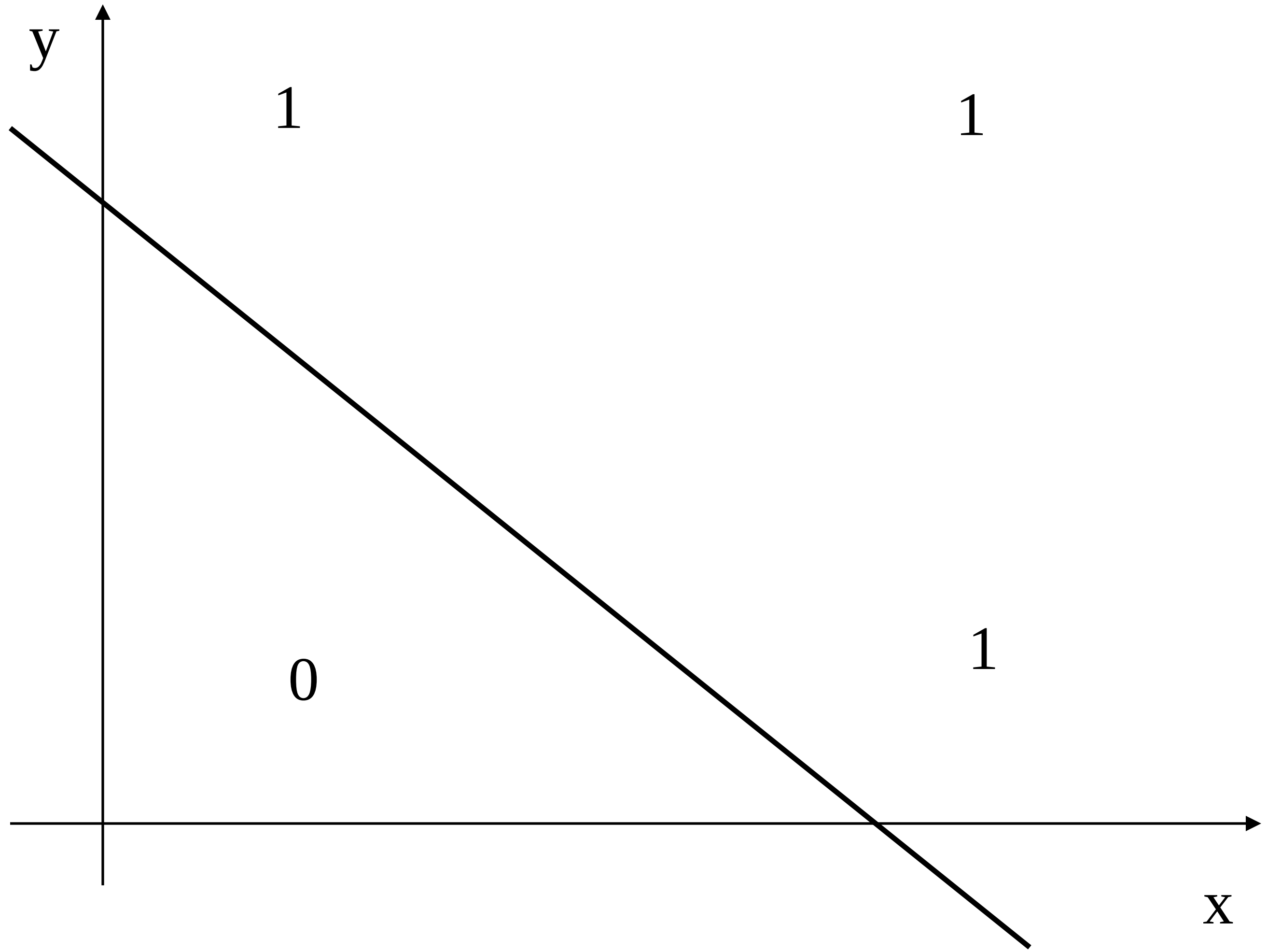
x	y	AND
0	0	0
0	1	0
1	0	0
1	1	1


y
0
1
0
0
x

Example Problem

- Logical OR
- Linearly separable?

x	y	OR
0	0	0
0	1	1
1	0	1
1	1	1

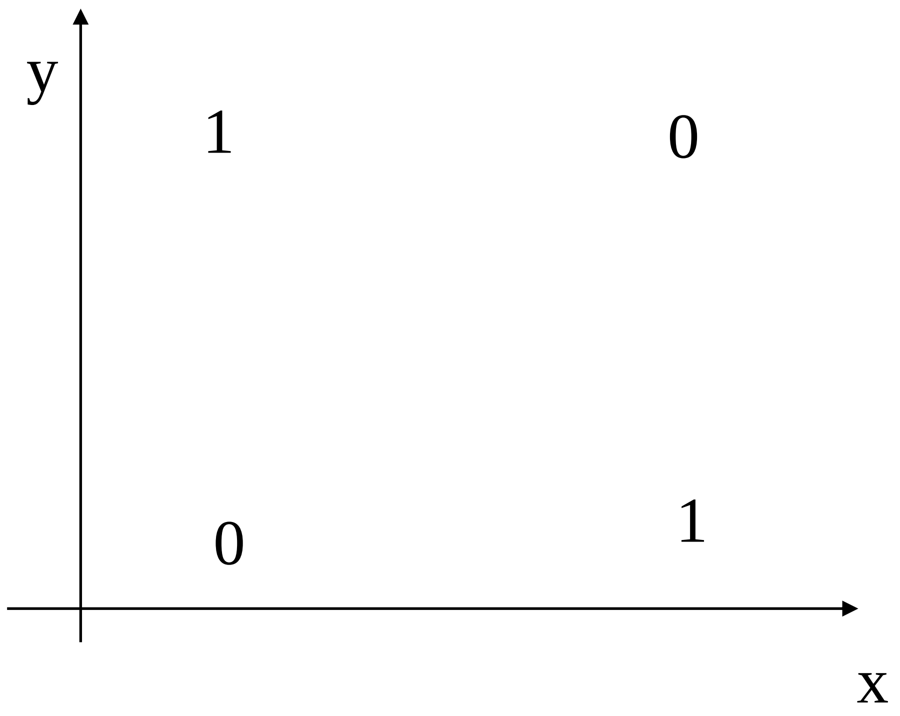

y
1
1
1
0
x

Example Problem

- Logical XOR
- Linearly separable?

x	y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Logical XOR



XOR (and many other problems like it) are not linearly separable.

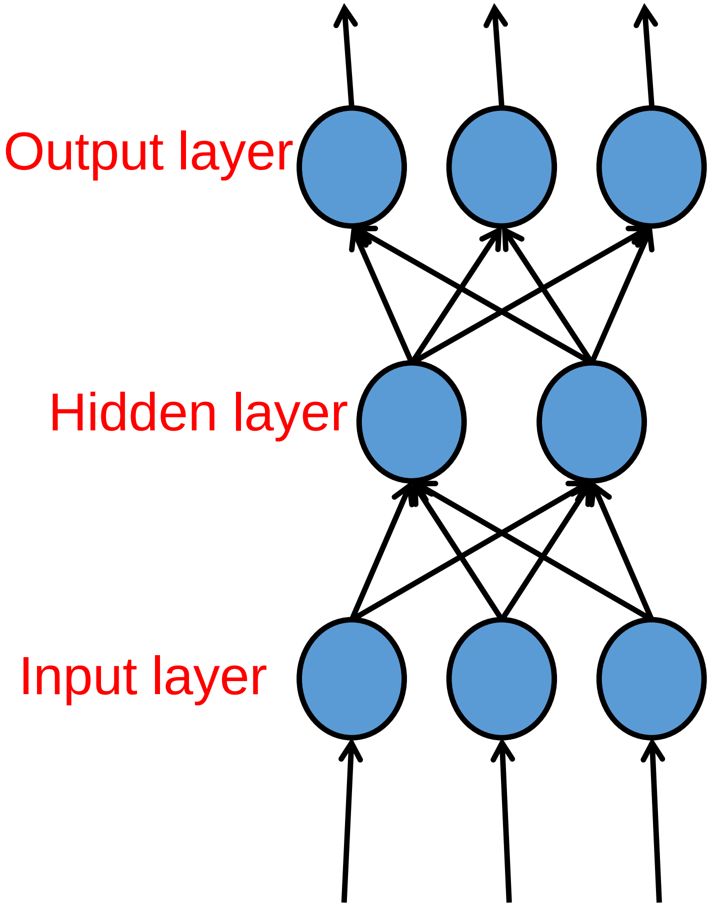
This scuppered research into neural networks for 20 years

Artificial Neural Networks

- Solution: combine neurons into a network
- By combining many neurons into a network can solve a wider range of more complex problems
  - Feedforward, multilayer networks
  - Recurrent (feedback) networks
  - Dynamic neural networks
- More powerful (in general), but harder to train

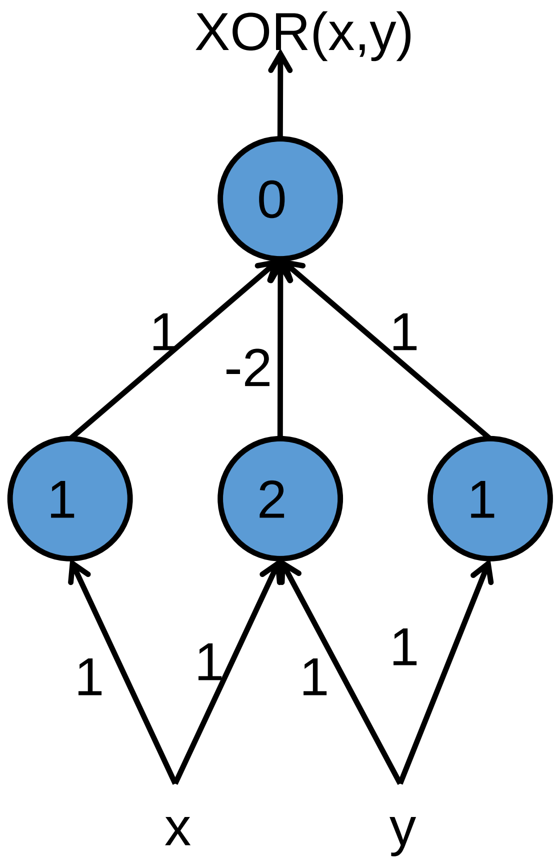
Multilayer Feedforward Networks

- Comprised of *layers* of simple neurons, each layer feeds into the next
- Can calculate any arbitrary function
- Well understood and can be trained using *backpropagation*
- Now applied to just about any machine learning problem



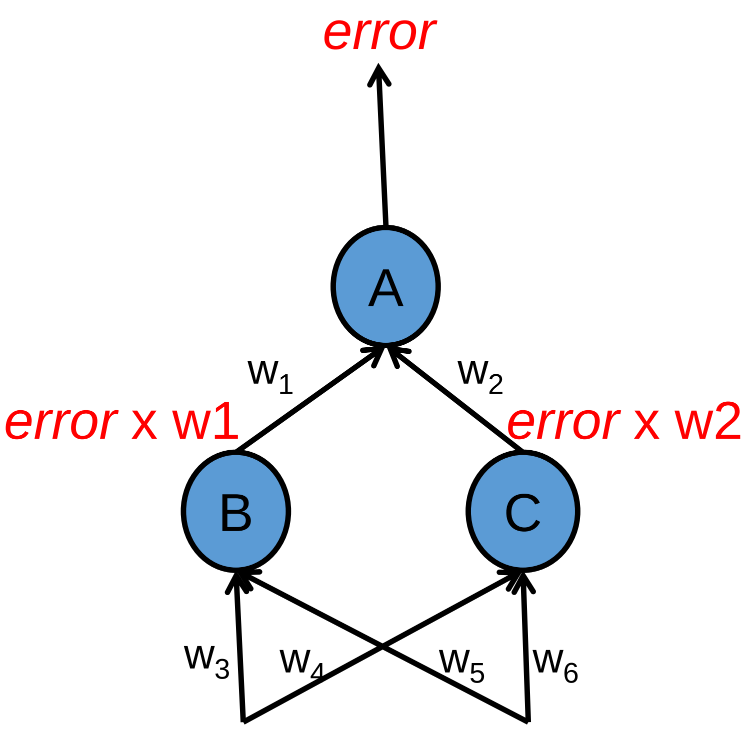
Multilayer Perceptron Example

- This 2-layer perceptron can solve the non-linearly seperable XOR problem.

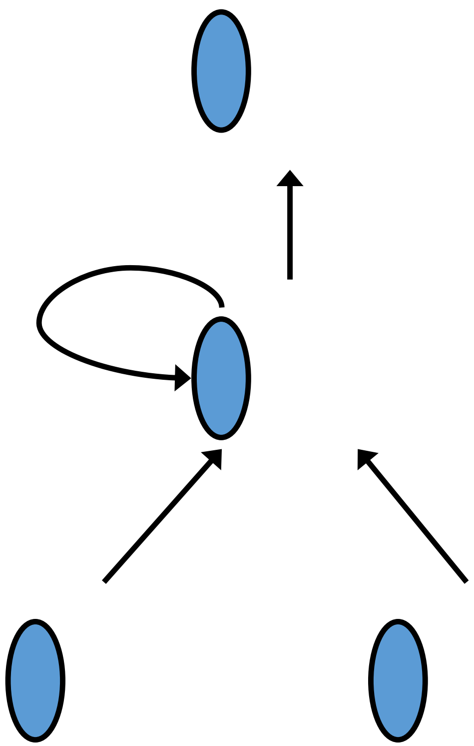


Backpropagation

- Invented in 1986, and made neural nets popular again.
- Problem:
  - Can use *error* to adjust input weights w_1 and w_2 for unit A
  - But what error do we use to adjust $w_{3\text{-}6}$ for the hidden layers?
- Solution:
  - 'backpropagate' the error to update the hidden weights
- en.wikipedia.org/wiki/Backpropagation

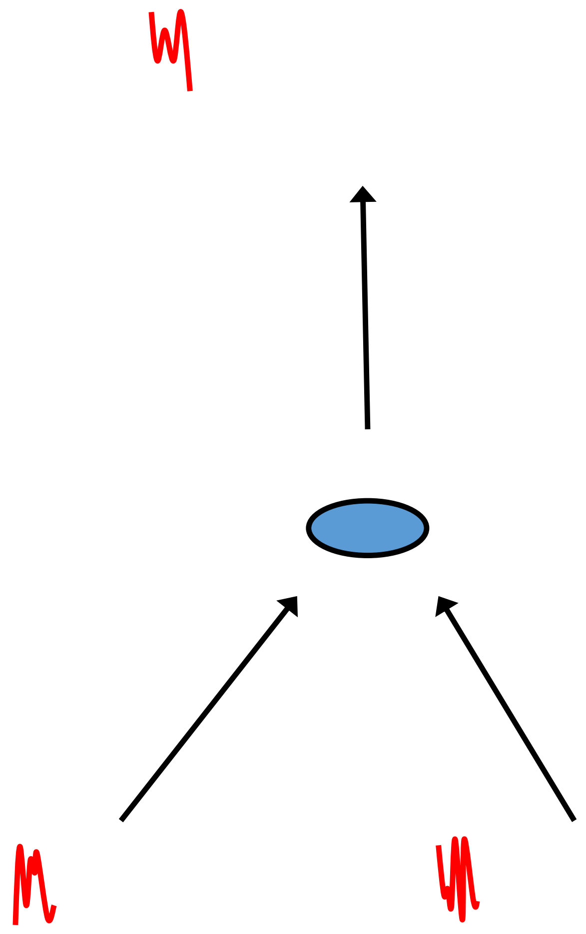


Elman Net



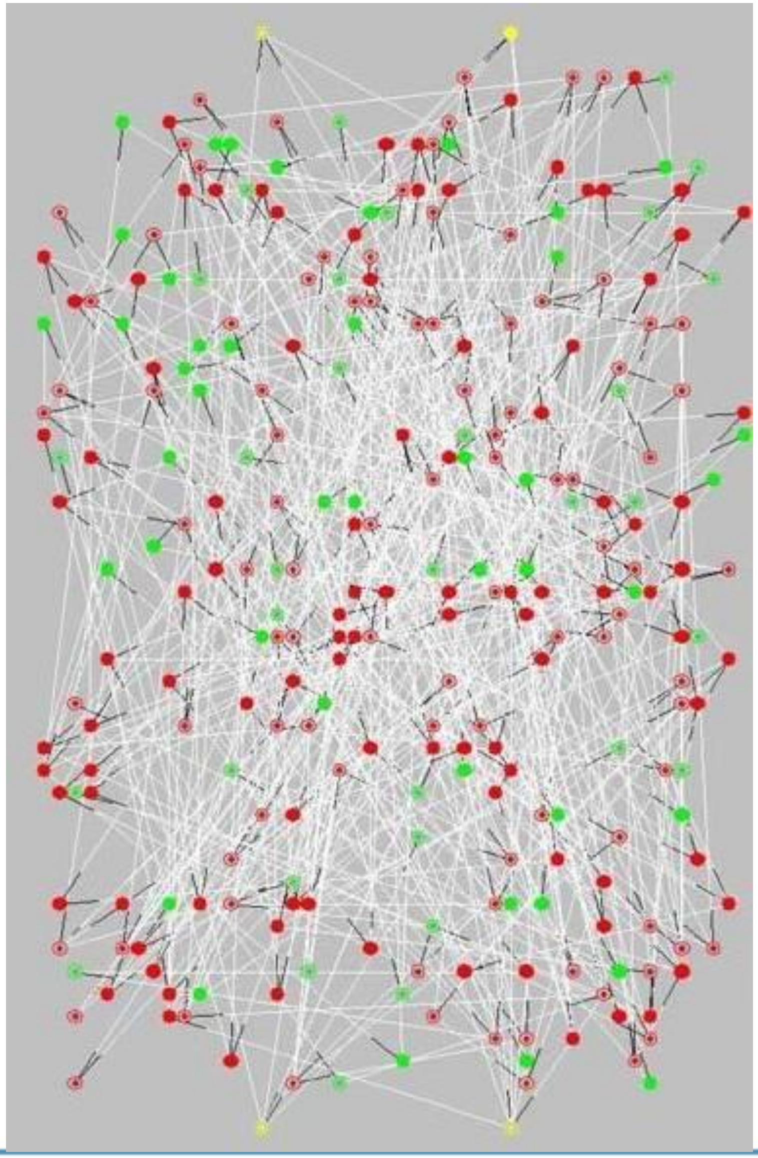
- Very simple *recursive* (feedback) network
- Previous state of neuron is fed back in at next time step
- Allows for simple memory or state
- Can distinguish order in sequences of inputs
- *Eg* simple natural language processing
  - *'dog is dead'*
  - *'is dog dead'*
- Or predicting time series
  - Stock markets
  - Weather
  - Sales

Dynamic (Real Time) Neural Network



- Continuous real inputs, activation equations and outputs
- Can be used for real time control problems
  - Robotics
  - Chemical plants
  - Jet engines

Recurrent Dynamic Neural Nets



- A lot more like the real brains that control our intelligent behaviour
- Continuous, real-time inputs and outputs (eg robot motors and sensors)
- Very complex dynamics – don't know how to make them do what we want
- Cannot reliably train them – so use evolution

Summary

- Biological Brains
- Artificial Neurons
- The Perceptron
- Learning
- Limits of Learning
- Artificial Neural Networks

Stretch Break!

Second Half (NEXT WEEK)

- Human Learning
- Machine Learning
  - Supervised Learning
  - Examples
- Data Mining
  - Examples

Thank you