Neural Networks

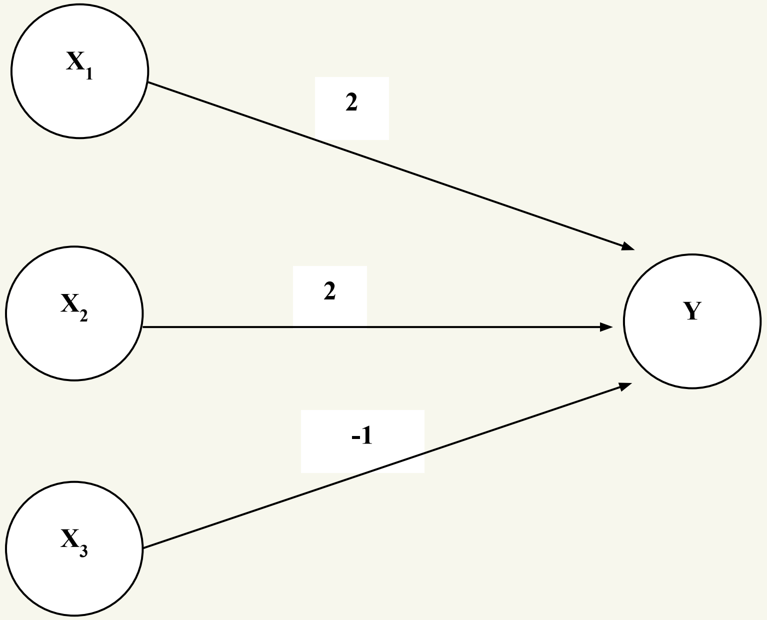
Moin Mostakim

The First Neural Network

McCulloch and Pitts produced the first neural network in 1943

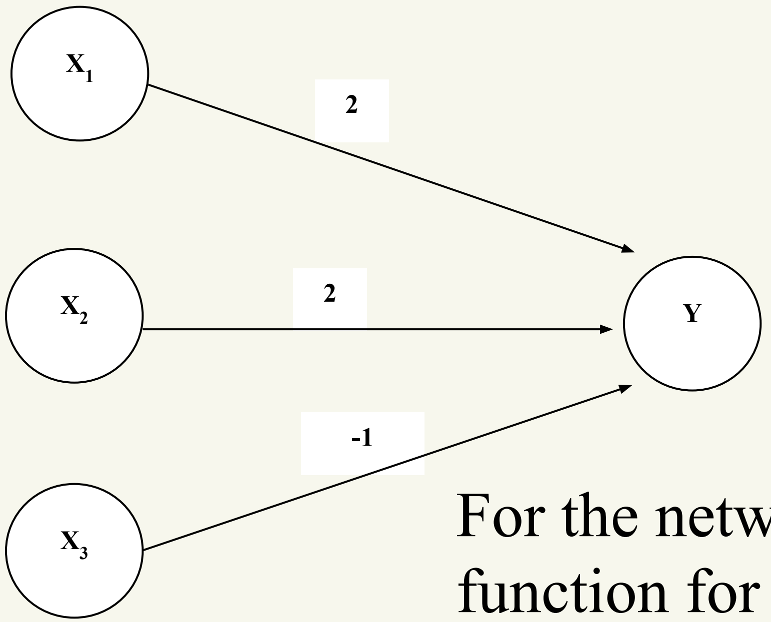
Many of the principles can still be seen in neural networks of today

The First Neural Network



The activation of a neuron is binary. That is, the neuron either fires (activation of one) or does not fire (activation of zero).

The First Neural Network



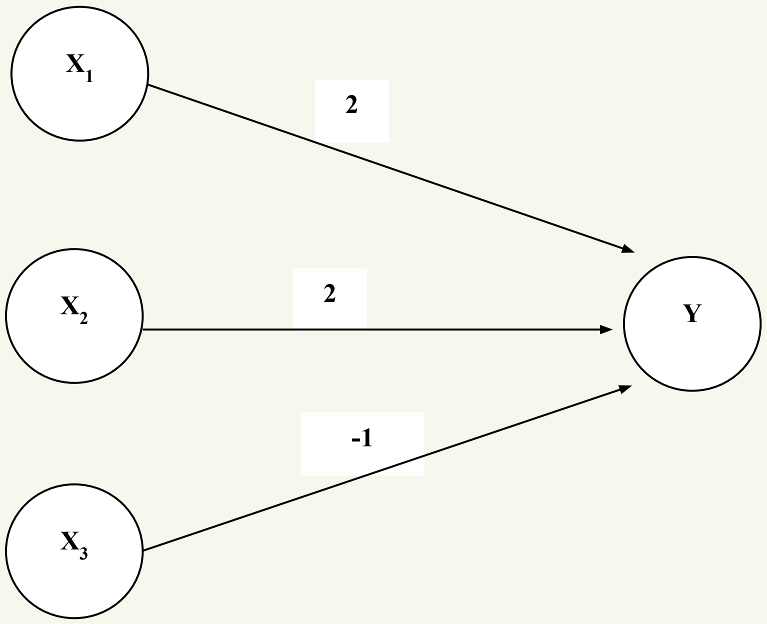
For the network shown here the activation function for unit Y is

$$f(y\_in) = 1, \text{ if } y\_in >= \theta \text{ else } 0$$

where y_in is the total input signal received

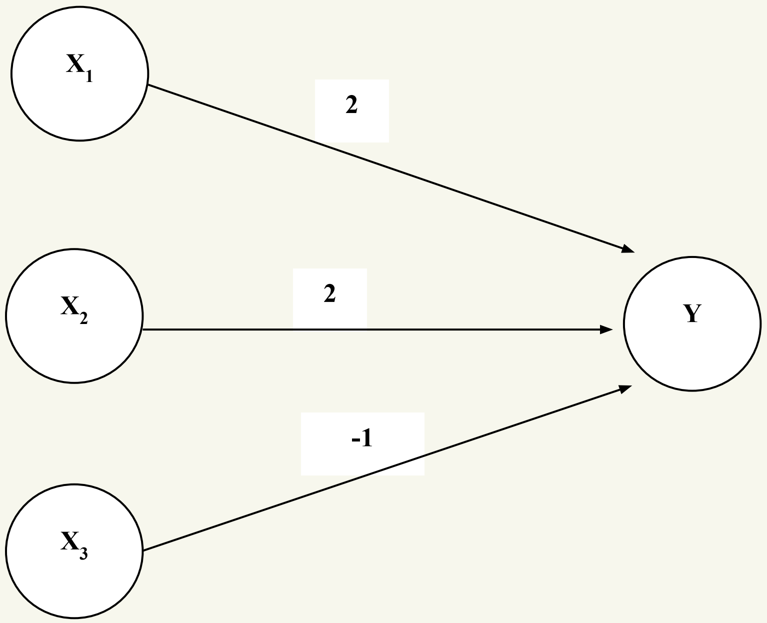
θ is the threshold for Y

The First Neural Network



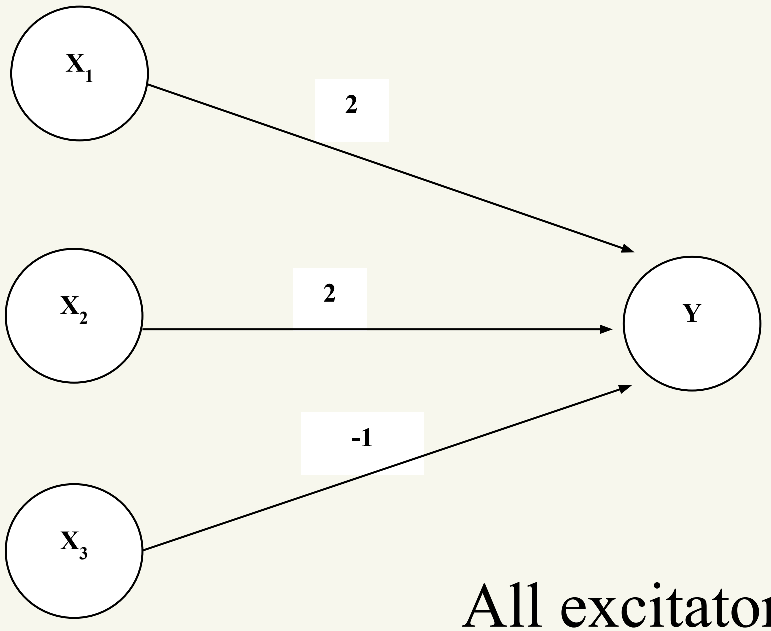
Neurons in a McCulloch-Pitts network are connected by directed, weighted paths

The First Neural Network



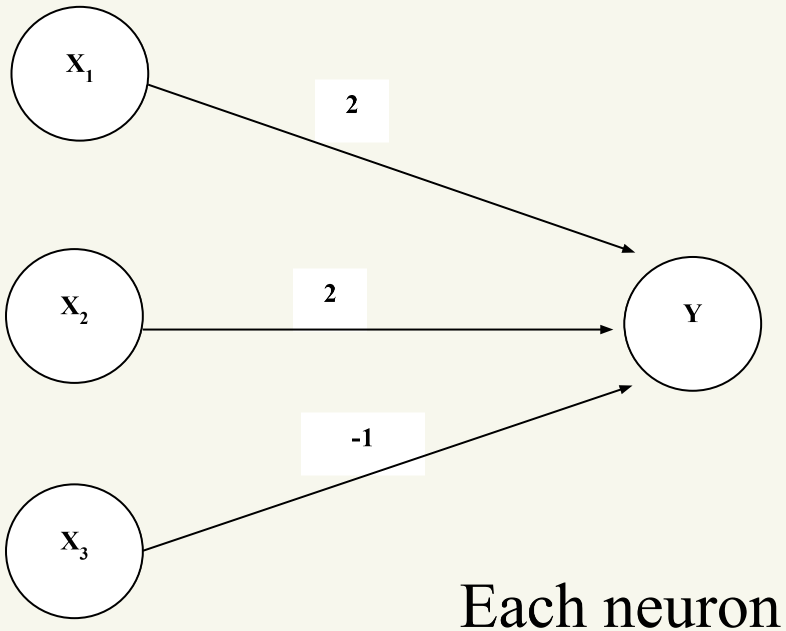
If the weight on a path is positive the path is excitatory, otherwise it is inhibitory

The First Neural Network



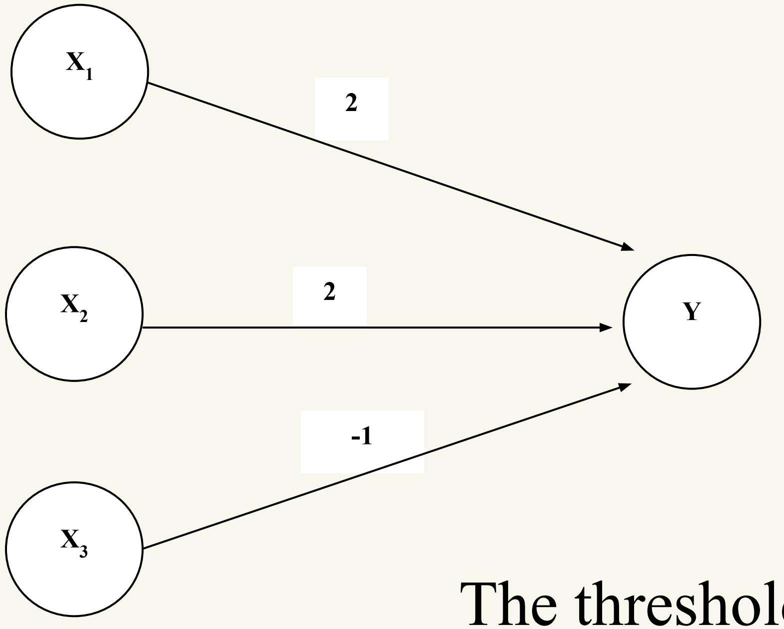
All excitatory connections into a particular neuron have the same weight, although different weighted connections can be input to different neurons

The First Neural Network



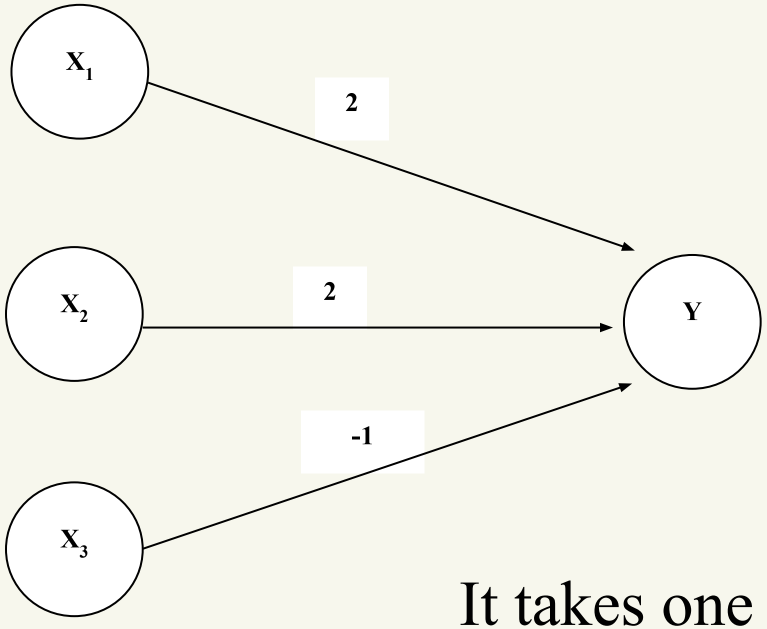
Each neuron has a fixed threshold. If the net input into the neuron is greater than the threshold, the neuron fires

The First Neural Network



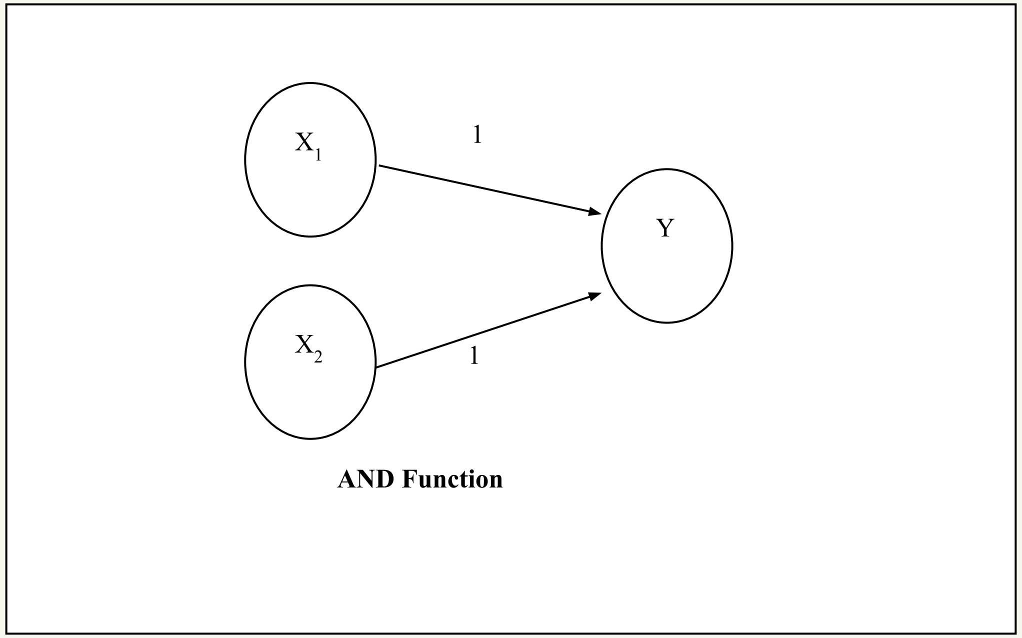
The threshold is set such that any non-zero inhibitory input will prevent the neuron from firing

The First Neural Network



It takes one time step for a signal to pass over one connection.

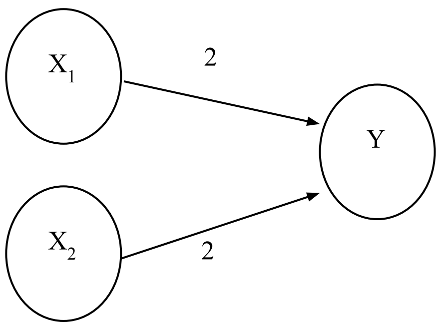
The First Neural Network



AND		
X1	**X2**	**Y**
1	1	1
1	0	0
0	1	0
0	0	0

Threshold(Y) = 2

The First Neural Network

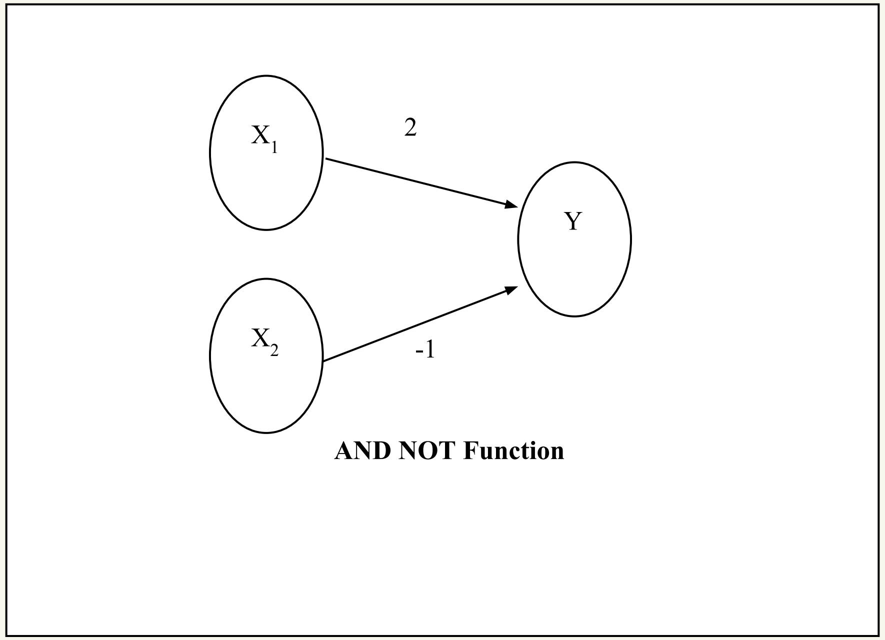


OR Function

$$\text{Threshold}(Y) = 2$$

OR		
X1	**X2**	**Y**
1	1	1
1	0	1
0	1	1
0	0	0

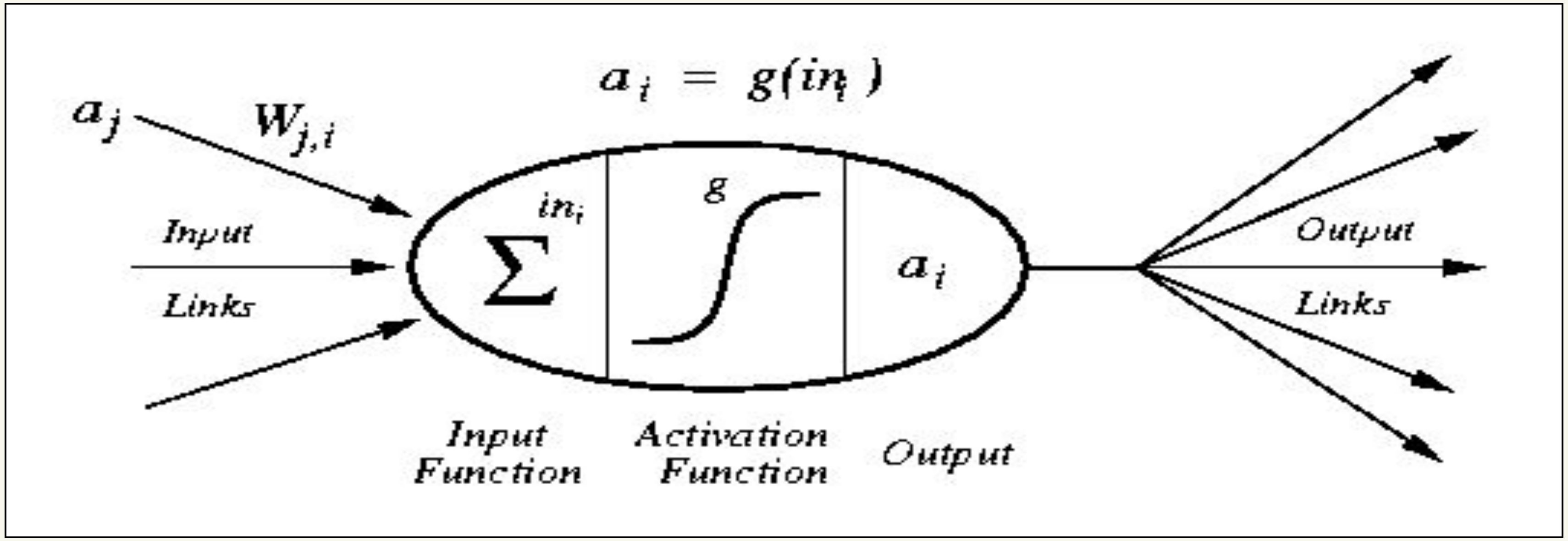
The First Neural Network



$$\text{Threshold}(Y) = 2$$

AND NOT		
X1	**X2**	**Y**
1	1	0
1	0	1
0	1	0
0	0	0

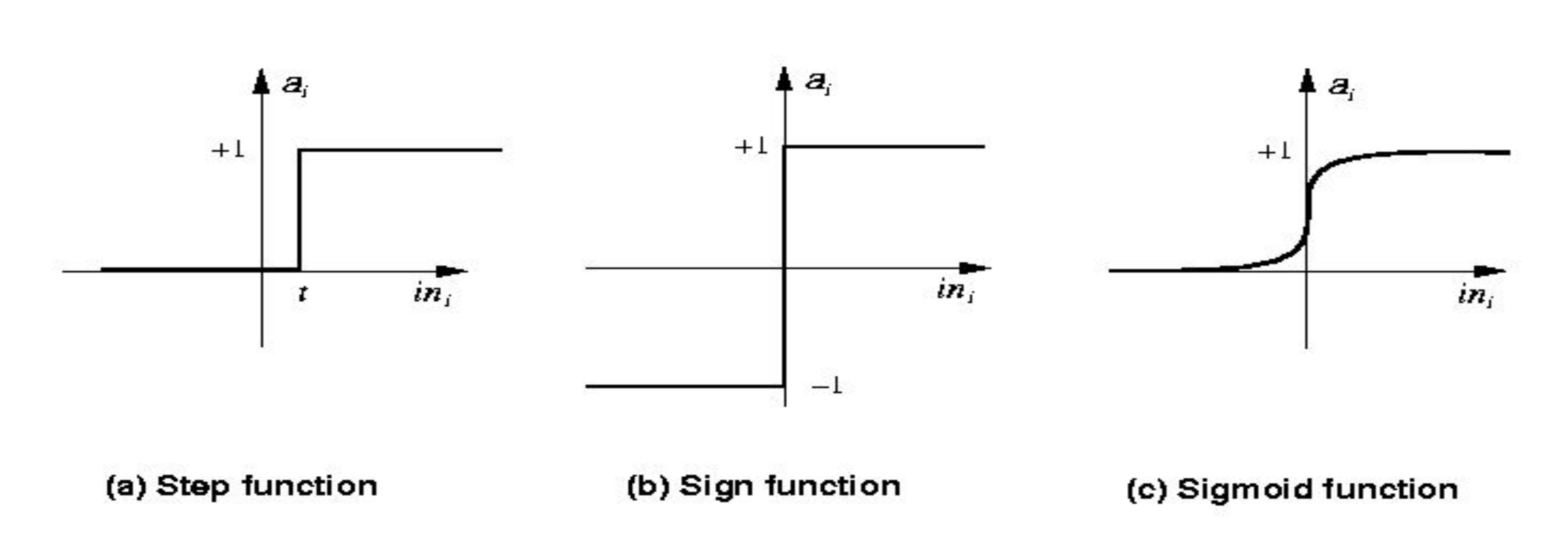
Modelling a Neuron



$$in_i = \sum_j W_{j,i} a_j$$

- **a_j :Activation value of unit j**
- **$w_{j,I}$:Weight on the link from unit j to unit i**
- **in_I :Weighted sum of inputs to unit i**
- **a_I :Activation value of unit i**
- **g :Activation function**

Activation Functions



- $Step_t(x) = 1$ if $x >= t$, else 0
- $Sign(x) = +1$ if $x >= 0$, else -1
- $Sigmoid(x) = 1/(1+e^{-x})$
- Identity Function

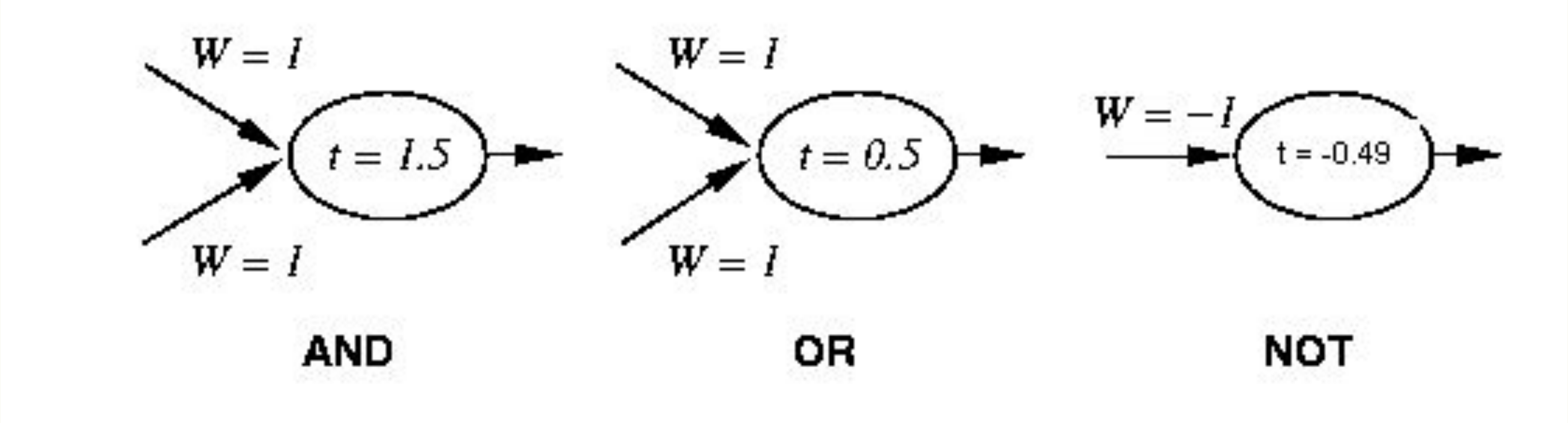
Simple Networks

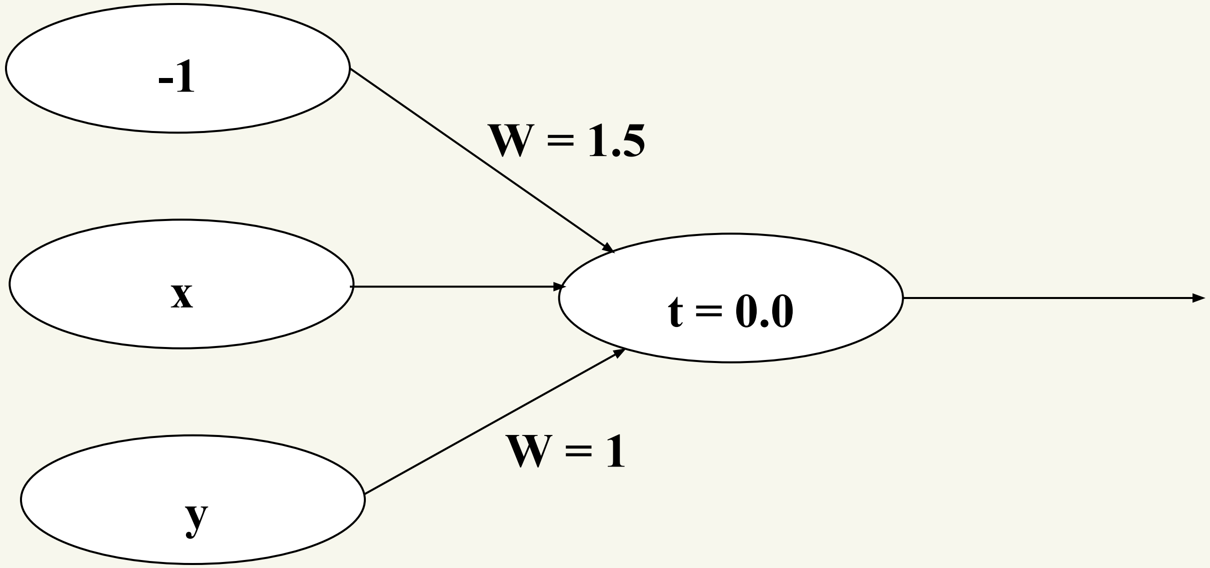
	AND			
Input 1	0	0	1	1
Input 2	0	1	0	1
Output	0	0	0	1

	OR			
Input 1	0	0	1	1
Input 2	0	1	0	1
Output	0	1	1	1

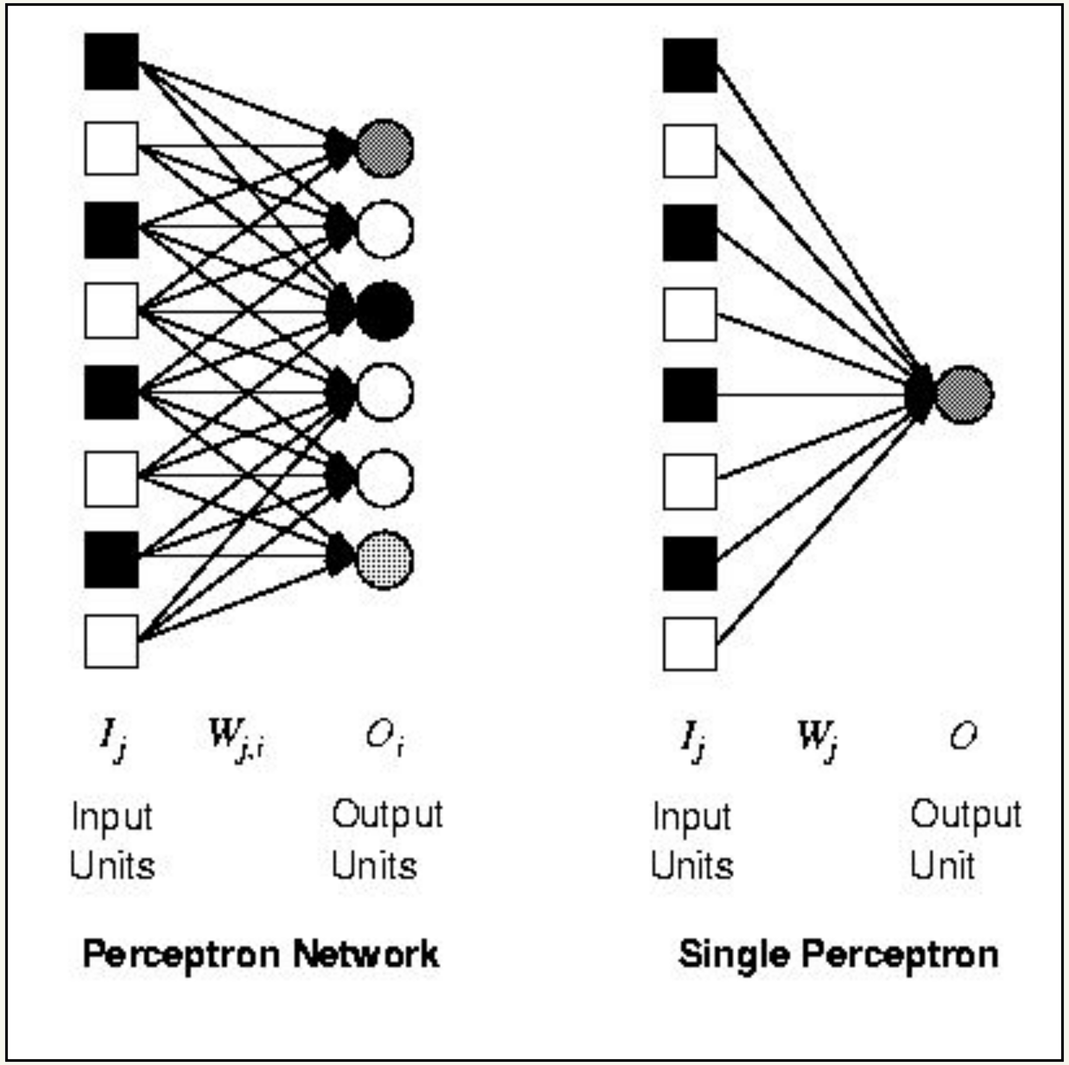
	NOT	
Input 1	0	1
Input 2		
Output	1	0

Simple Networks



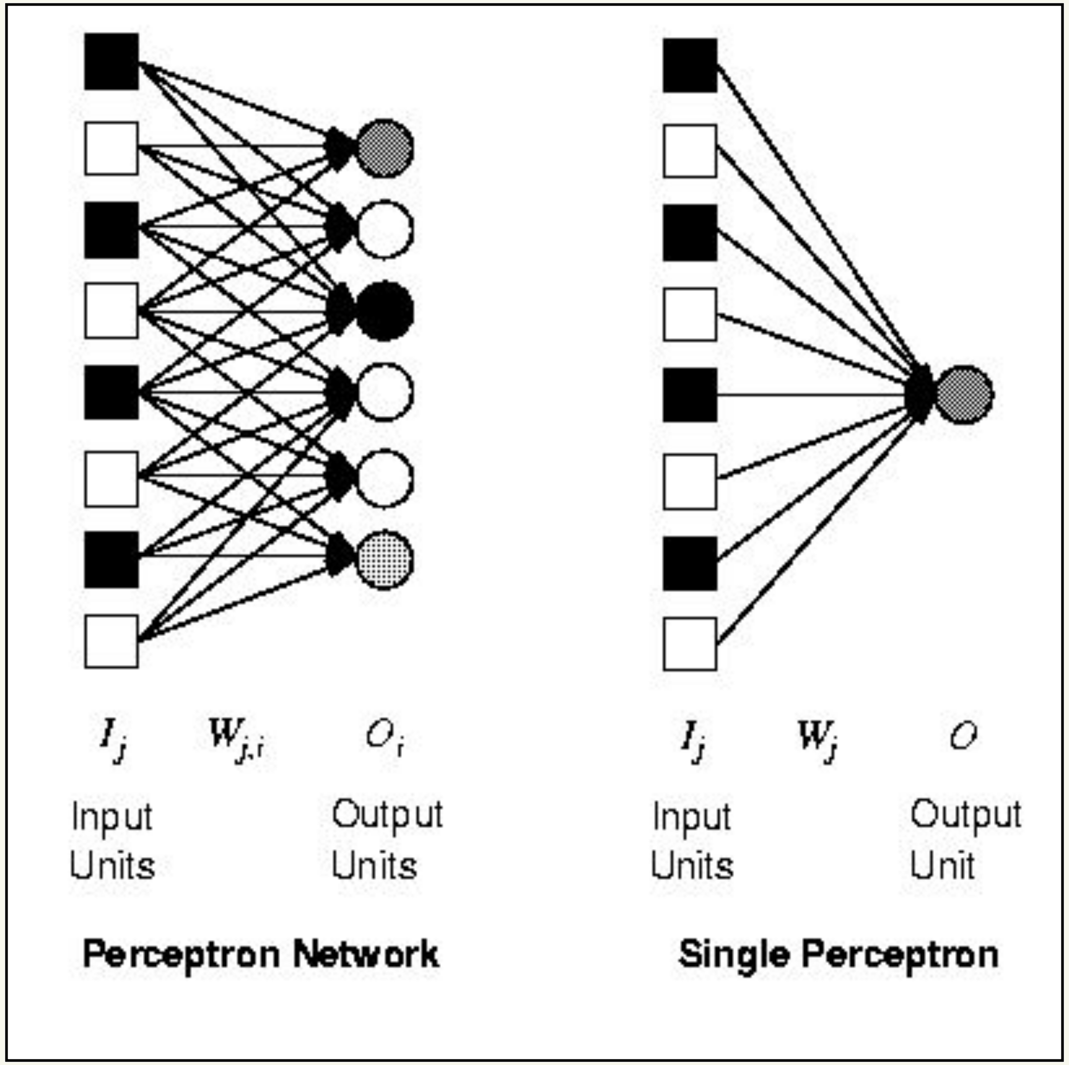


Perceptron



- **Synonym for Single-Layer, Feed-Forward Network**
- **First Studied in the 50's**
- **Other networks were known about but the perceptron was the only one capable of learning and thus all research was concentrated in this area**

Perceptron



- **A single weight only affects one output so we can restrict our investigations to a model as shown on the right**
- **Notation can be simpler, i.e.**

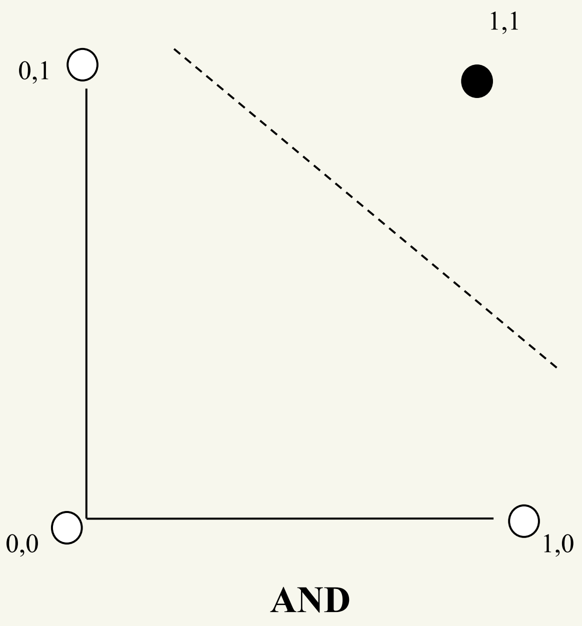
$$O = Step_0 \sum_j WjIj$$

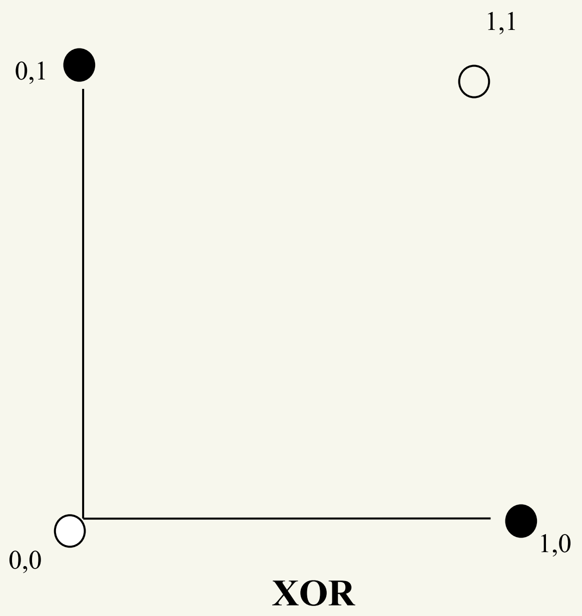
What can perceptron's represent?

	AND			
Input 1	0	0	1	1
Input 2	0	1	0	1
Output	0	0	0	1

	XOR			
Input 1	0	0	1	1
Input 2	0	1	0	1
Output	0	1	1	0

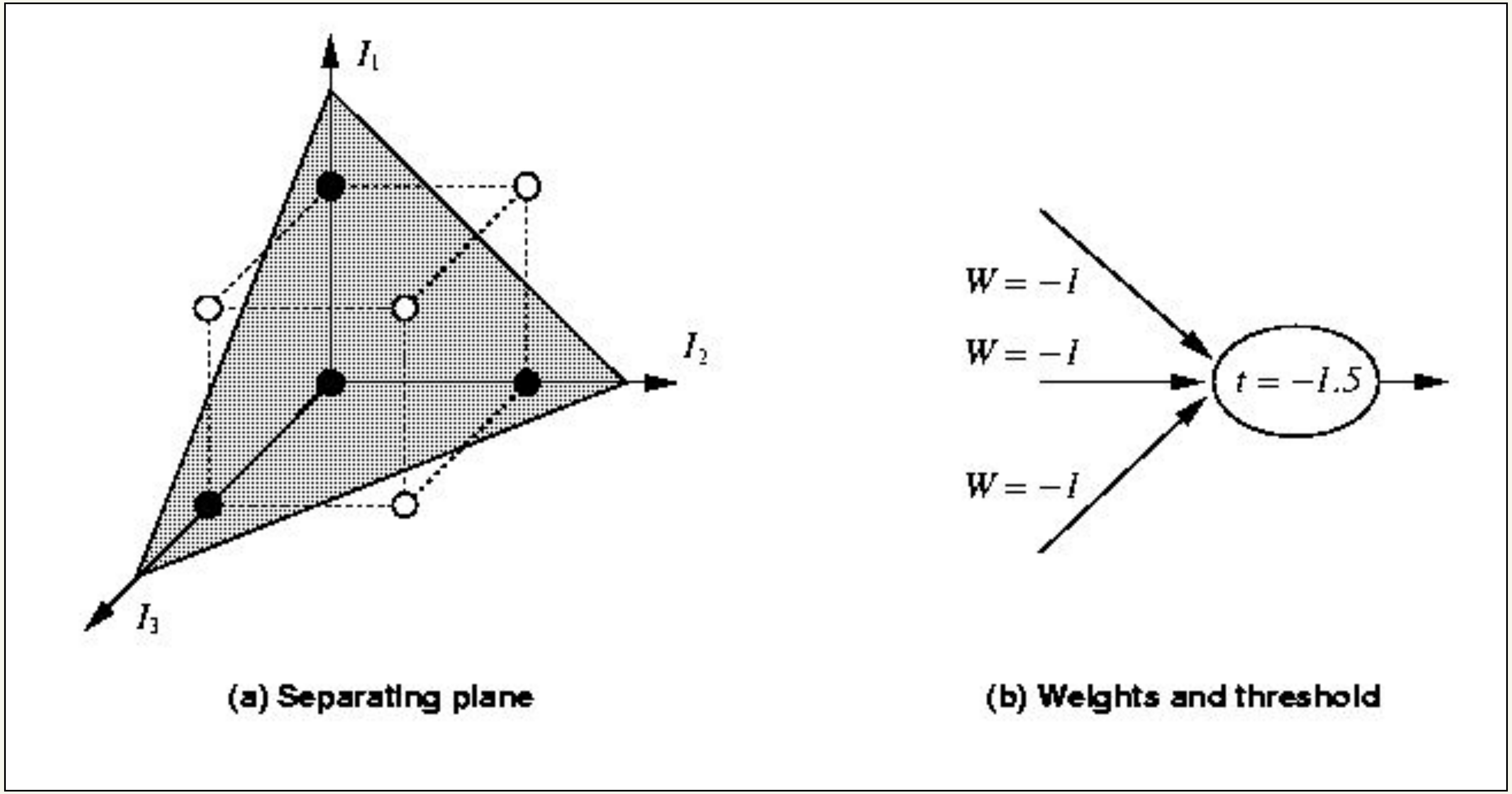
What can perceptrons represent?





- **Functions which can be separated in this way are called *Linearly Separable***
- **Only linearly Separable functions can be represented by a perceptron**

What can perceptrons represent?



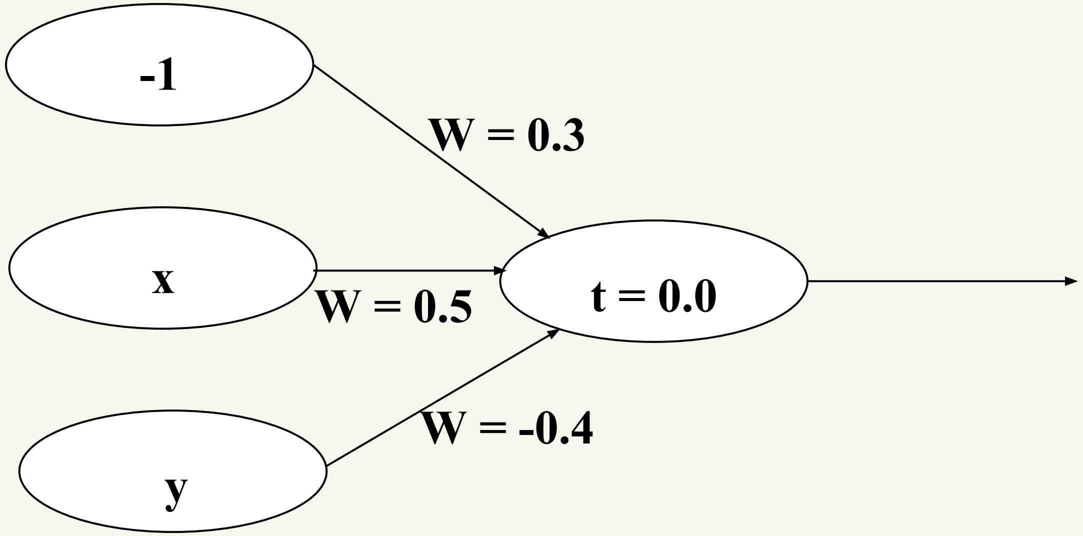
Linear Separability is also possible in more than 3 dimensions – but it is harder to visualise

Training a perceptron

Aim

	AND			
Input 1	0	0	1	1
Input 2	0	1	0	1
Output	0	0	0	1

Training a perceptrons



I_1	I_2	I_3	Summation	Output
-1	0	0	(-1*0.3) + (0*0.5) + (0*-0.4) = -0.3	0
-1	0	1	(-1*0.3) + (0*0.5) + (1*-0.4) = -0.7	0
-1	1	0	(-1*0.3) + (1*0.5) + (0*-0.4) = 0.2	1
-1	1	1	(-1*0.3) + (1*0.5) + (1*-0.4) = -0.2	0

Learning

While epoch produces an error

Present network with next inputs from epoch

Err = T – O

If Err <> 0 then

$$W_j = W_j + LR * I_j * Err$$

End If

End While

Learning

While epoch produces an error

Present network with next inputs from epoch

Err = T – O

If Err <> 0 then

$W_j = W_j + LR * I_j * Err$

End If

End While

Epoch : Presentation of the entire training set to the neural network.

In the case of the AND function an epoch consists of four sets of inputs being presented to the network (i.e. [0,0], [0,1], [1,0], [1,1])

Learning

While epoch produces an error

Present network with next inputs from epoch

Err = T – O

If Err <> 0 then

$W_j = W_j + LR * I_j * Err$

End If

End While

Training Value, T : When we are training a network we not only present it with the input but also with a value that we require the network to produce. For example, if we present the network with [1,1] for the AND function the training value will be 1

Learning

While epoch produces an error

> **Present network with next inputs from epoch**
>
> **Err = T – O**
>
> **If Err <> 0 then**
>
> > $W_j = W_j + LR * I_j * Err$
>
> **End If**

End While

<u>Error, Err</u> : The error value is the amount by which the value output by the network differs from the training value. For example, if we required the network to output 0 and it output a 1, then Err = -1

Learning

While epoch produces an error

Present network with next inputs from epoch

Err = T – O

If Err <> 0 then

$W_j = W_j + LR * I_j * Err$

End If

End While

Output from Neuron, O : The output value from the neuron

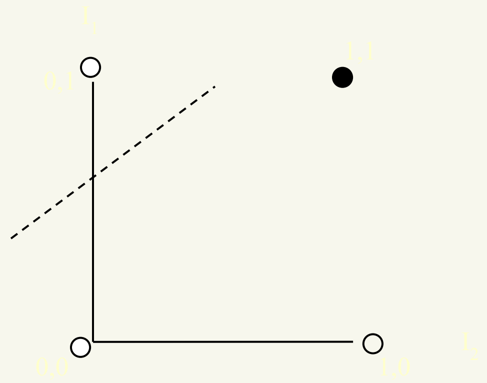
Ij : Inputs being presented to the neuron

Wj : Weight from input neuron (I_j) to the output neuron

LR : The learning rate. This dictates how quickly the network converges. It is set by a matter of experimentation. It is typically 0.1

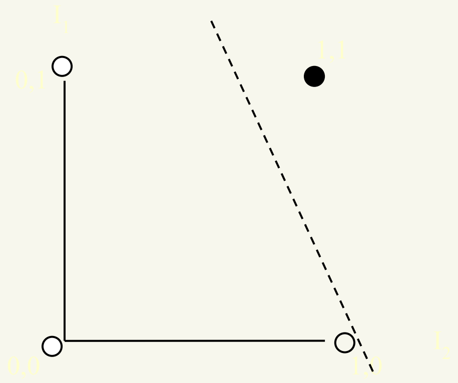
Learning

After First Epoch

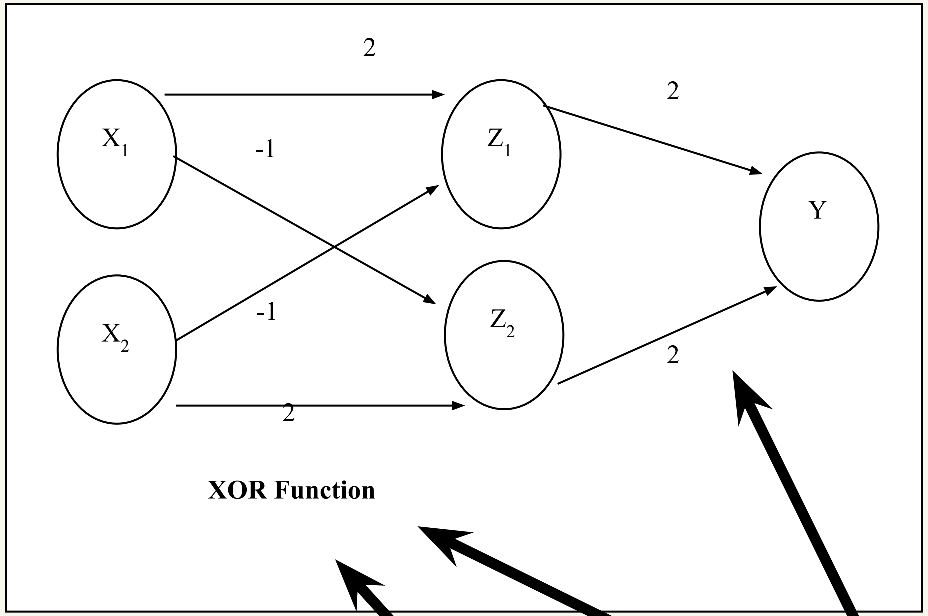


> **<u>Note</u>**
> I_1 point = W_0/W_1
> I_2 point = W_0/W_2

At Convergence



The First Neural Network



XOR		
X1	**X2**	**Y**
1	1	0
1	0	1
0	1	1
0	0	0

$$X_1 \text{ XOR } X_2 = (X_1 \text{ AND NOT } X_2) \text{ OR } (X_2 \text{ AND NOT } X_1)$$

The First Neural Network

If we touch something cold we perceive heat

If we keep touching something cold we will perceive cold

If we touch something hot we will perceive heat

The First Neural Networks

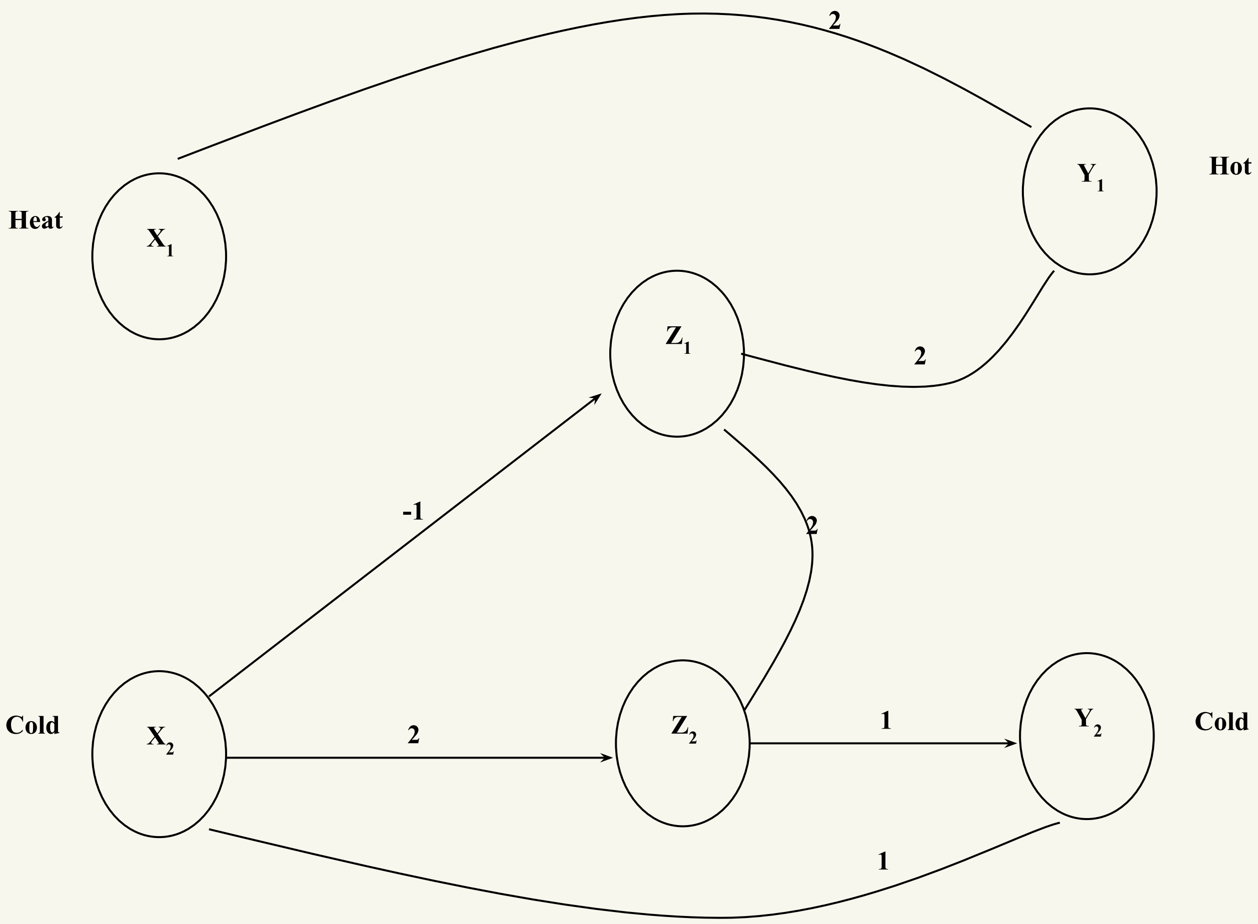
To model this we will assume that time is discrete

If cold is applied for one time step then heat will be perceived

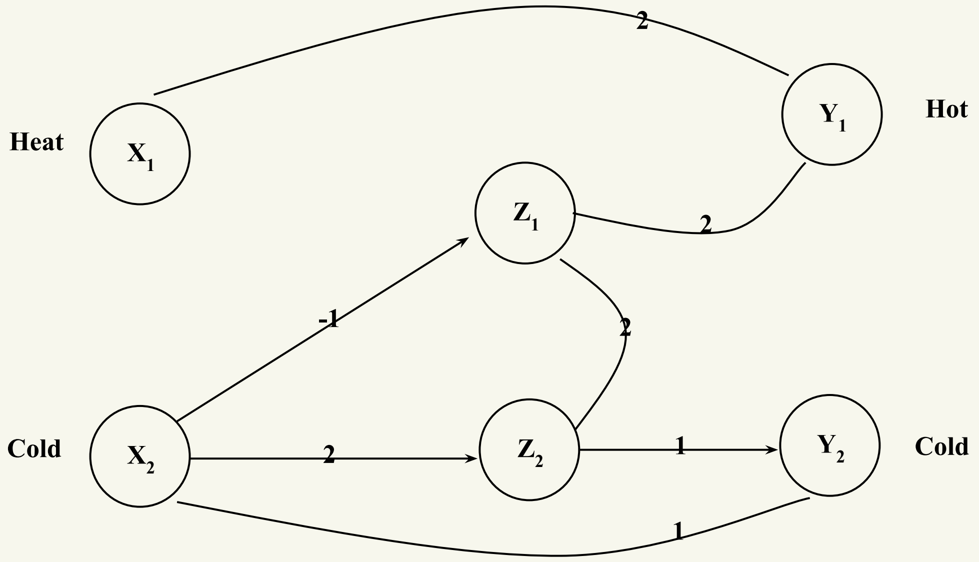
If a cold stimulus is applied for two time steps then cold will be perceived

If heat is applied then we should perceive heat

The First Neural Networks



The First Neural Networks



- It takes time for the stimulus (applied at X_1 and X_2) to make its way to Y_1 and Y_2 where we perceive either heat or cold

- At t(0), we apply a stimulus to X_1 and X_2
- At t(1) we can update Z_1, Z_2 and Y_1
- At t(2) we can perceive a stimulus at Y_2
- At t(2+n) the network is fully functional

The First Neural Network

We want the system to perceive cold if a cold stimulus is applied for two time steps

$$Y_2(t) = X_2(t-2)\ AND\ X_2(t-1)$$

$X_2(t-2)$	$X_2(t-1)$	$Y_2(t)$
1	1	1
1	0	0
0	1	0
0	0	0

The First Neural Network

We want the system to perceive heat if either a hot stimulus is applied or a cold stimulus is applied (for one time step) and then removed

$$Y_1(t) = [\ X_1(t-1)\ ] \text{ OR } [\ X_2(t-3) \text{ AND NOT } X_2(t-2)\ ]$$

X2(t – 3)	X2(t – 2)	AND NOT	X1(t – 1)	OR
1	1	0	1	1
1	0	1	1	1
0	1	0	1	1
0	0	0	1	1
1	1	0	0	0
1	0	1	0	1
0	1	0	0	0
0	0	0	0	0

The First Neural Network

The network shows

$$Y_1(t) = X_1(t-1) \text{ OR } Z_1(t-1)$$

$$Z_1(t-1) = Z_2(t-2) \text{ AND NOT } X_2(t-2)$$

$$Z_2(t-2) = X_2(t-3)$$

Substituting, we get

$$Y_1(t) = [\, X_1(t-1) \,] \text{ OR } [\, X_2(t-3) \text{ AND NOT } X_2(t-2) \,]$$

which is the same as our original requirements

The First Neural Network

You can confirm that Y_2 works correctly

You can also check it works on the spreadsheet

Threshold
2

Time	Heat (X1)	Cold (X2)	Z1	Z2	Hot (Y1)	Cold (Y2)
0	0	1				
1	0	0	0	1		
2	0	0	1	0	0	0
3	0	0			1	0

Time	Heat (X1)	Cold (X2)	Z1	Z2	Hot (Y1)	Cold (Y2)
0	0	1				
1	0	1	0	1		
2	0	0	0	1	0	1

Time	Heat (X1)	Cold (X2)	Z1	Z2	Hot (Y1)	Cold (Y2)
0	1	0				
1	1	0	0	0		
2	0	0	0	0	1	0

	X1	X2	Z1	Z2
Z1		-1		2
Z2		2		
Y1	2		2	
Y2		1		1