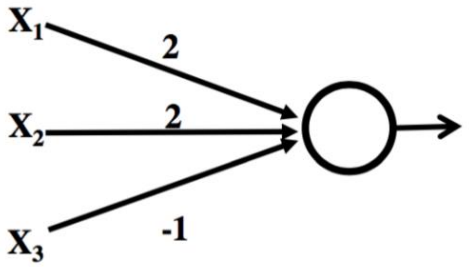
Neural Networks Fundamentals of AI (AE1FAI)

Slides created by Dr Rong Qu & Dr John Drake
Modified by Dr Huan Jin

NEURAL NETWORKS



McCulloch & Pitts (1943) are generally recognised as the designers of the first neural network

- Many simple units combine to give an increased computational power
- The idea of a threshold
- Many of their ideas still used today

Hebb (1949) developed the first learning rule

- McCulloch & Pitts network has fixed weights
- If two neurons were active at the same time the strength between them should be increased

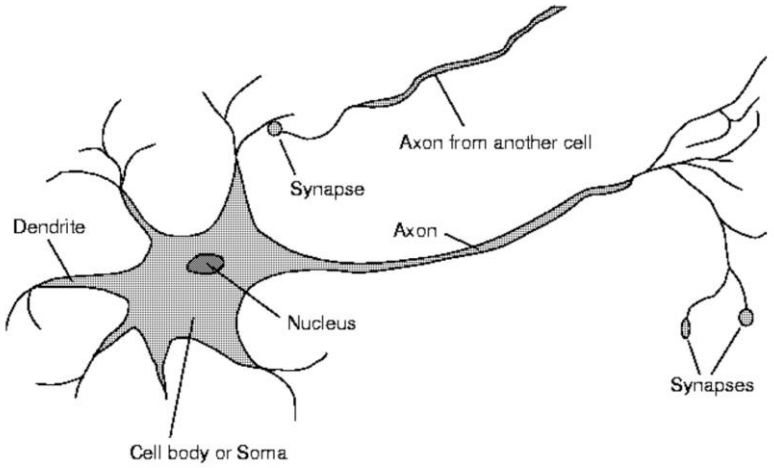
NEURAL NETWORKS

During the 50's and 60's

- Many researchers worked on the **perceptron**, leading to great excitement
- This model can be proved to converge to the correct weights: learning → thinking
- More powerful learning algorithm than Hebb

1969 saw the death of neural network research

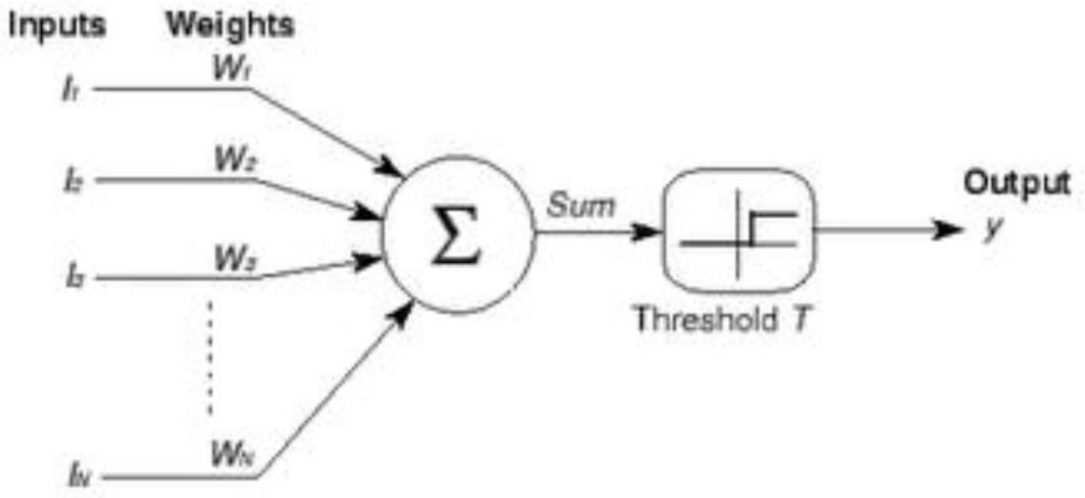
- **Minsky & Papert**
- Perceptron can't learn certain type of important functions
- Research of ANN went to decline for about 15 years



THE FIRST NEURAL NETWORKS

McCulloch and Pitts produced the first neural network in 1943

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

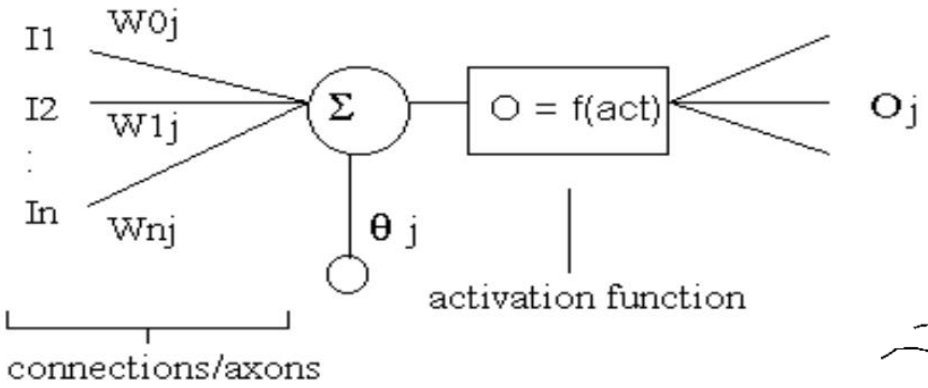


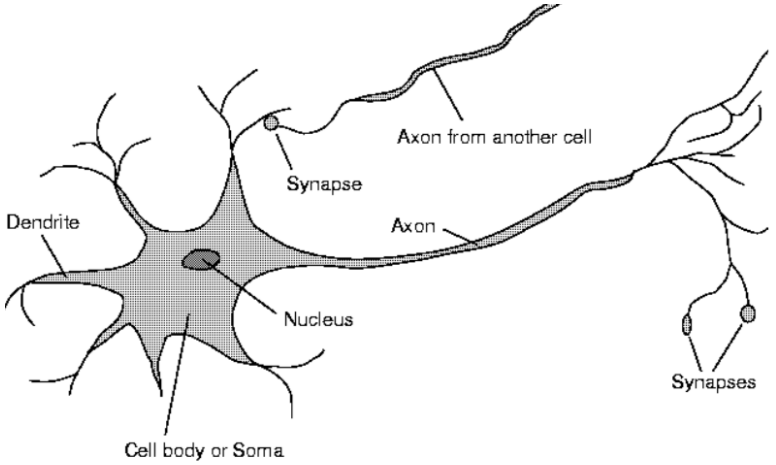
THE FIRST NEURAL NETWORKS

Consists of:

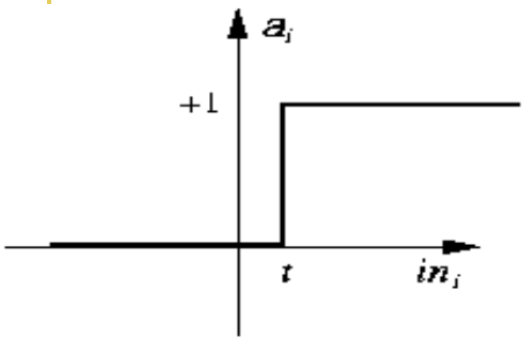
- A set of inputs - (dendrites)
- A set of resistances/weights – (synapses)
- A processing element - (neuron)
- A single output - (axon)

The activation of a neuron is **binary**. That is, the neuron either fires or does not fire (all or nothing)

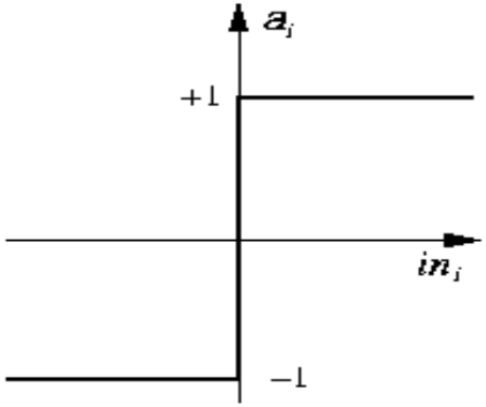




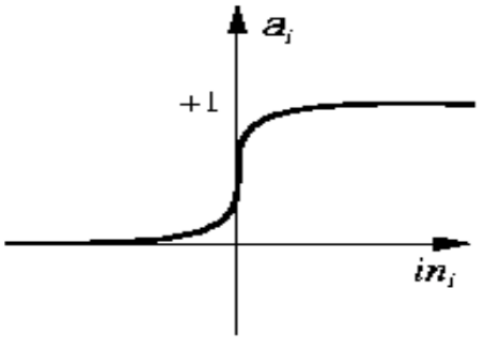
ACTIVATION FUNCTIONS



(a) Step function



(b) Sign function



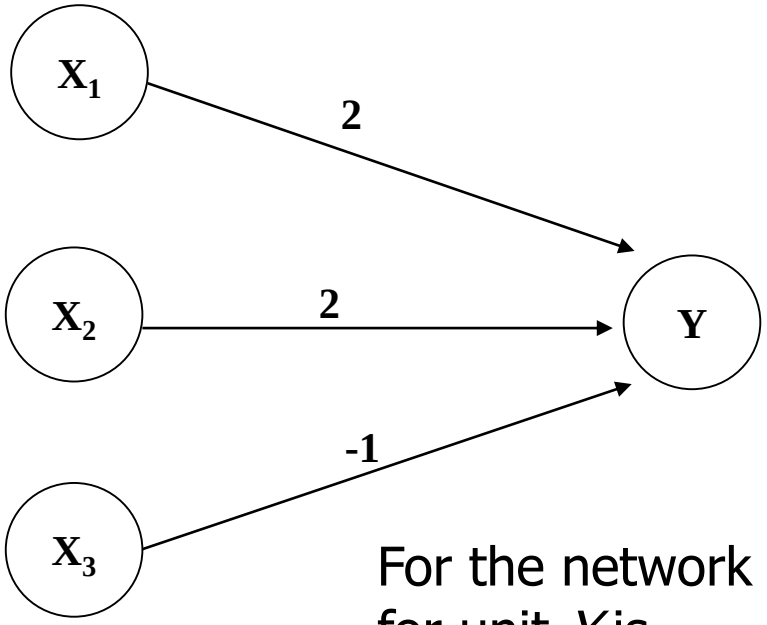
(c) Sigmoid function

$Step_t(x)$ = 1 if x >= t, else 0

Sign(x) = +1 if x >= 0, else –1

Sigmoid(x) = $1/(1+e^{-x})$

THE FIRST NEURAL NETWORKS



$y\_in = x_1 * 2 + x_2 * 2 + x_3 * (-1)$
Weighted Sum

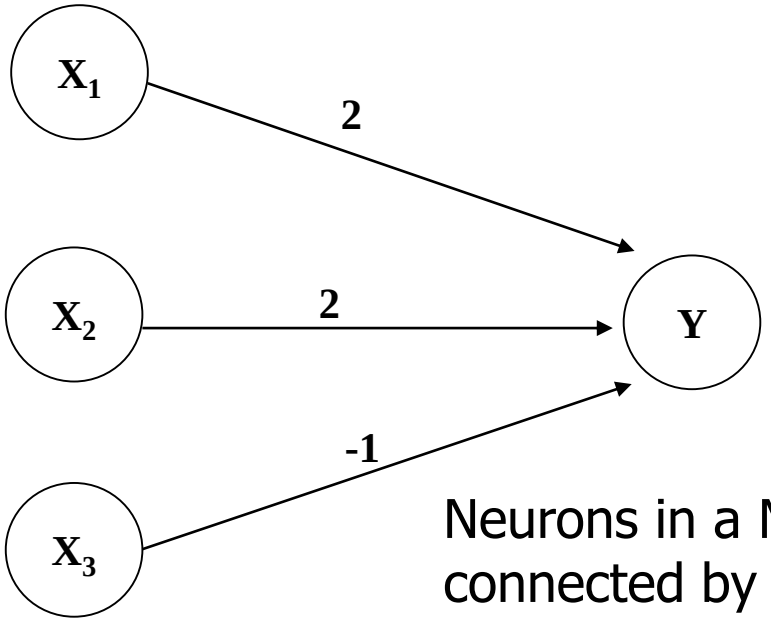
For the network shown here the activation function for unit *Y* is

f(y_in) = 1 (fire), if *y_in* >= θ; else 0 (not fire)

where *y_in* is the total input signal received (weighted sum);

θ is the **threshold** for *Y*

THE FIRST NEURAL NETWORKS

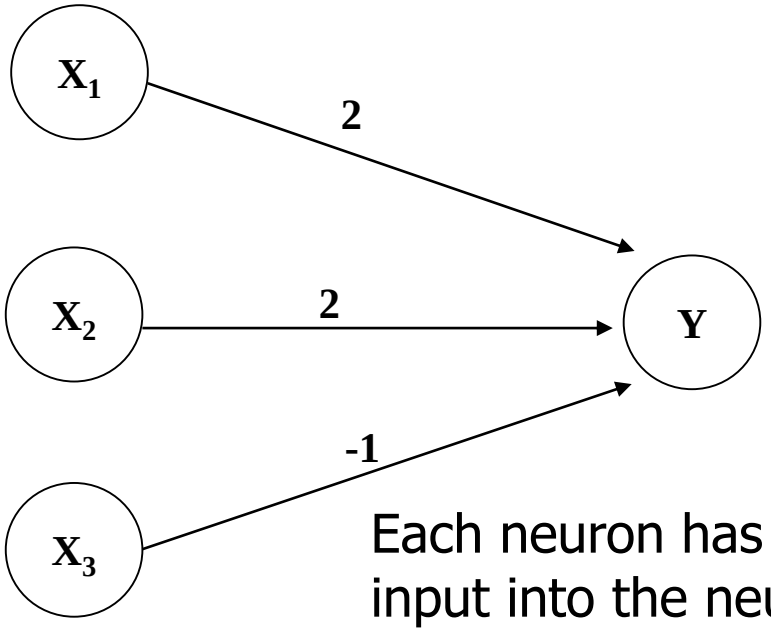


Neurons in a McCulloch-Pitts network are connected by directed, weighted paths

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

- x_1 and x_2 encourage the neuron to fire
- x_3 prevents the neuron from firing

THE FIRST NEURAL NETWORKS

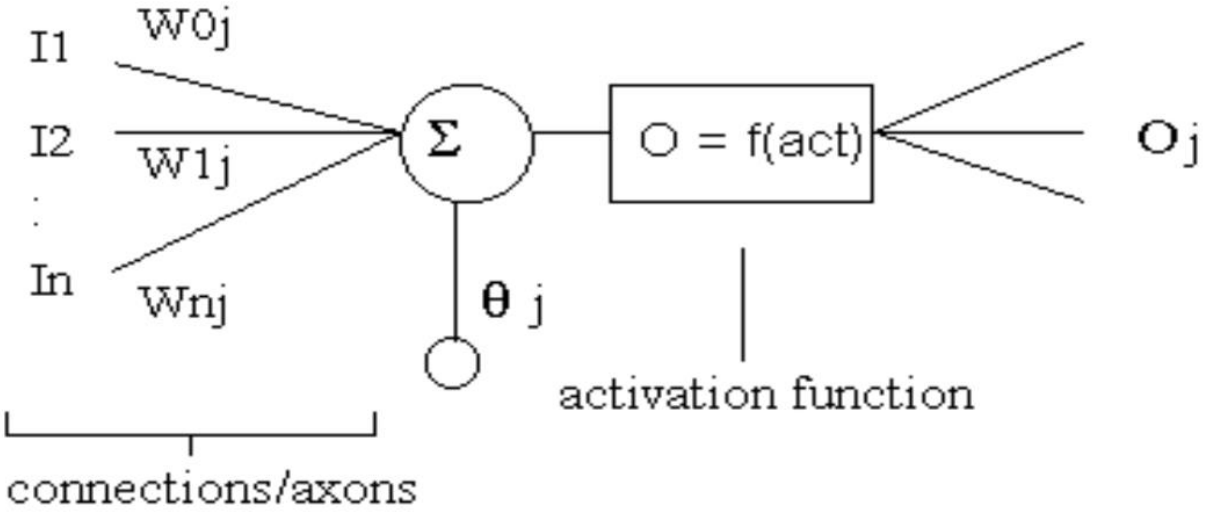


Each neuron has a fixed **threshold**. If the net input into the neuron is greater than or equal to the threshold, the neuron fires

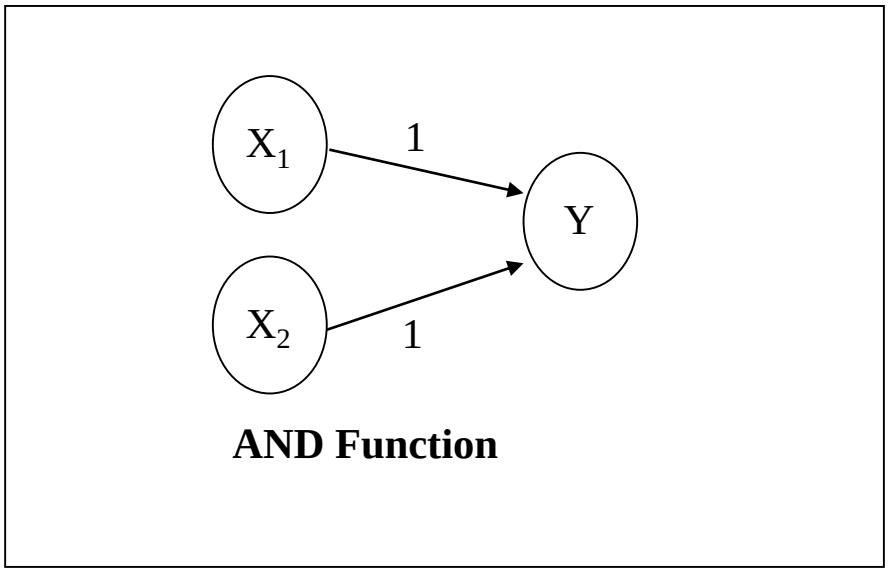
THE FIRST NEURAL NETWORKS

Using McCulloch-Pitts model we can model logic functions

Let's look at 4 logic functions



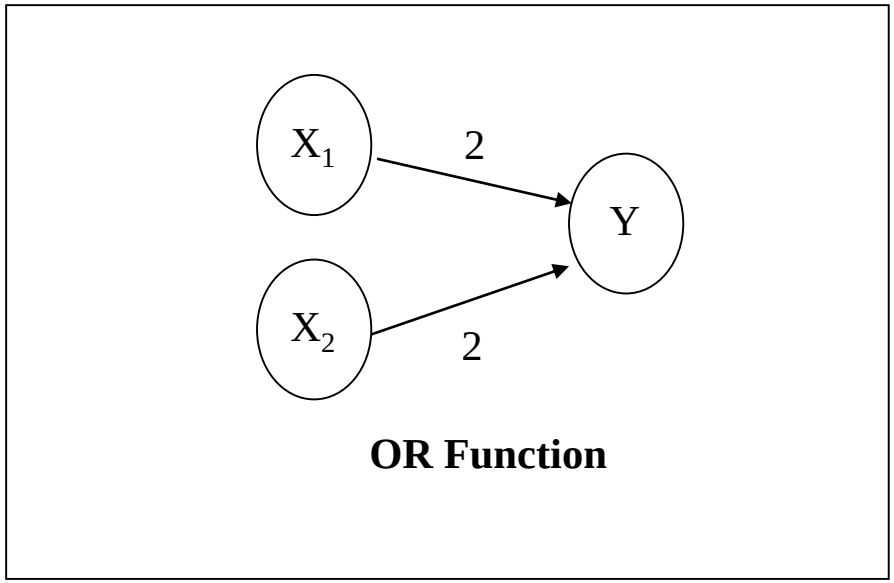
THE FIRST NEURAL NETWORKS



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

Threshold(Y) = 2

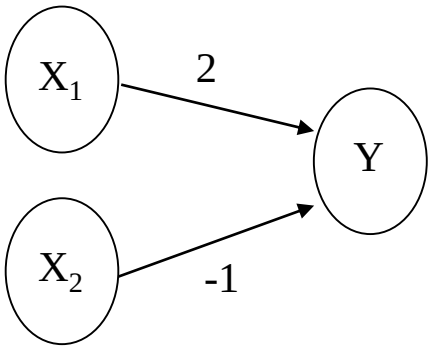
THE FIRST NEURAL NETWORKS



Threshold(*Y*) = 2

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

THE FIRST NEURAL NETWORKS



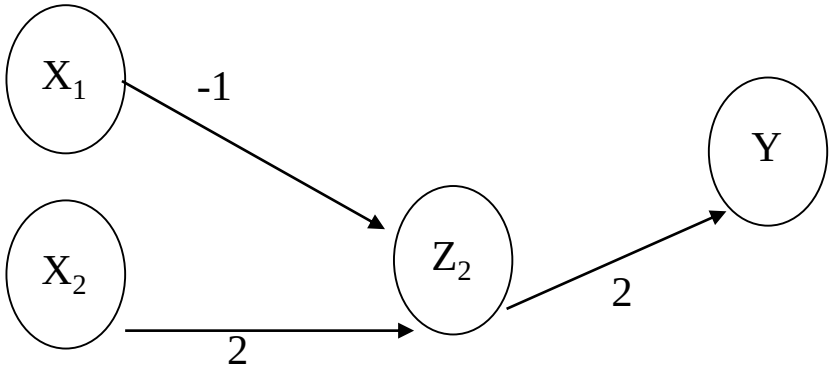
AND NOT Function

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

Threshold(*Y*) = 2

THE FIRST NEURAL NETWORKS

X_1 XOR X_2 = (X_1 AND NOT X_2) OR (X_2 AND NOT X_1)

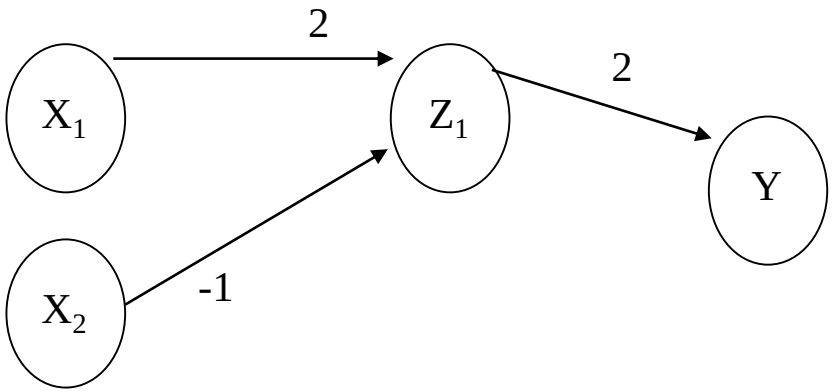


XOR		
X1	X2	Y
1	1	0
1	0	1
0	1	1
0	0	0

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

THE FIRST NEURAL NETWORKS

X_1 XOR X_2 = (X_1 AND NOT X_2) OR (X_2 AND NOT X_1)

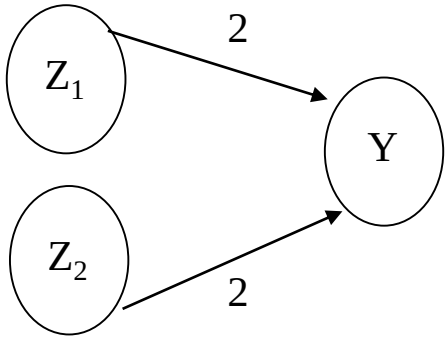


XOR		
X1	X2	Y
1	1	0
1	0	1
0	1	1
0	0	0

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

THE FIRST NEURAL NETWORKS

X_1 XOR X_2 = (X_1 AND NOT X_2) OR (X_2 AND NOT X_1)

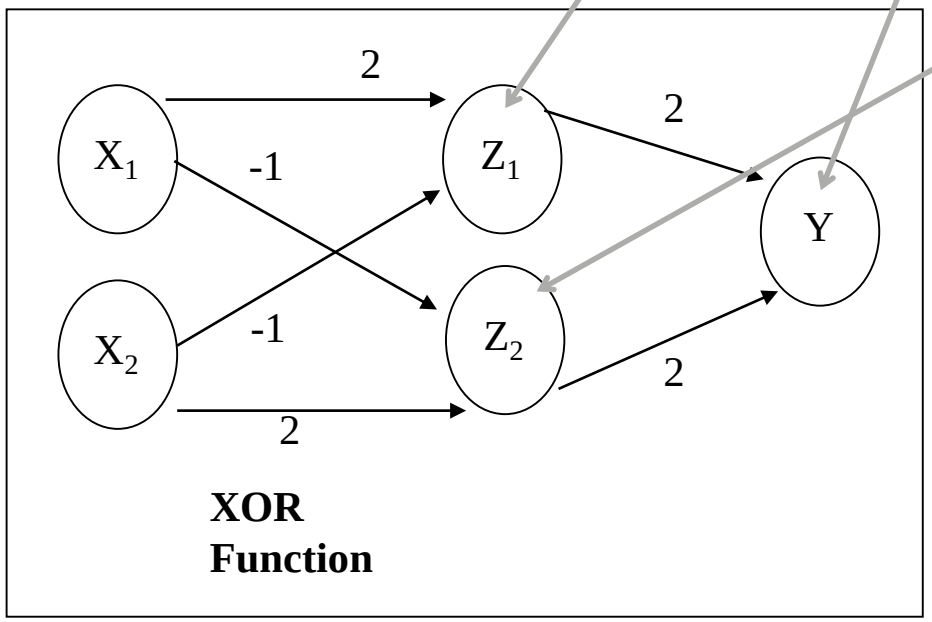


XOR		
X1	X2	Y
1	1	0
1	0	1
0	1	1
0	0	0

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

THE FIRST NEURAL NETWORKS

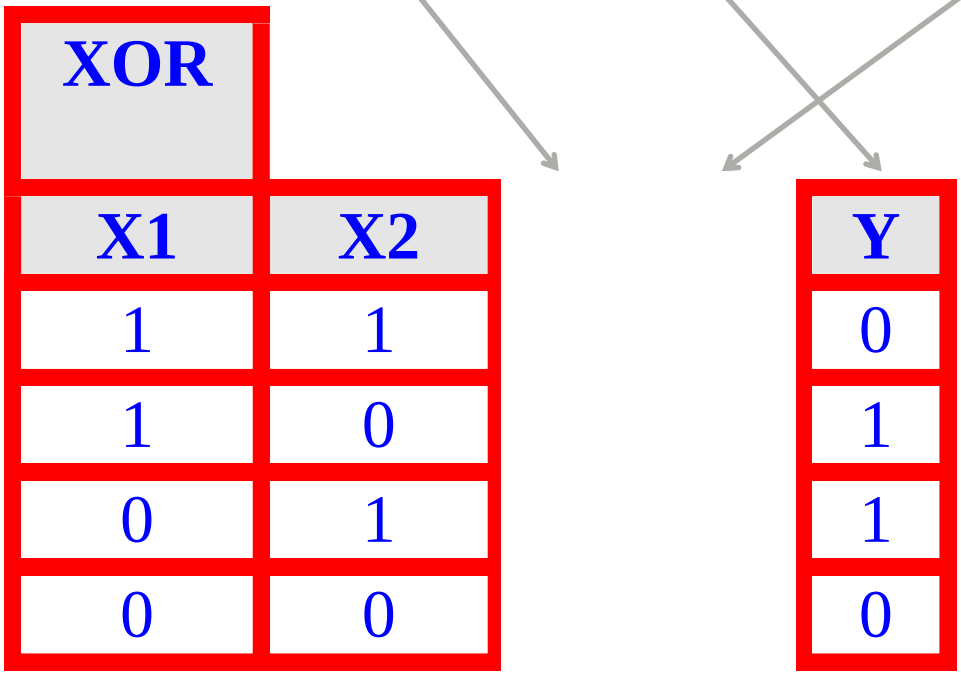
X_1 XOR X_2 = (X_1 AND NOT X_2) OR (X_2 AND NOT X_1)



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

THE FIRST NEURAL NETWORKS

X_1 XOR X_2 = (X_1 AND NOT X_2) OR (X_2 AND NOT X_1)



XOR	
X1	X2
1	1
1	0
0	1
0	0

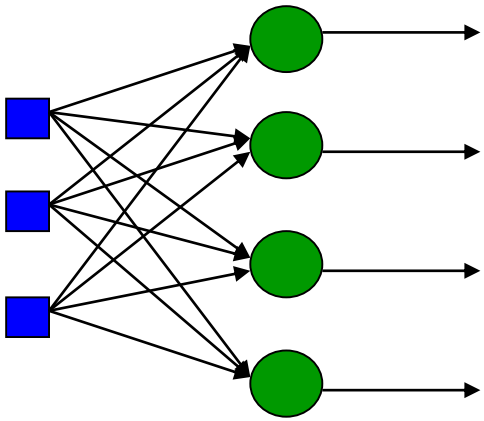
Y
0
1
1
0

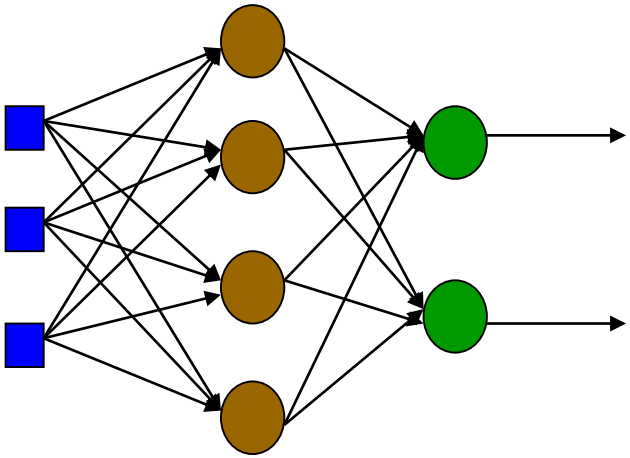
NEURAL NETWORK ARCHITECTURES

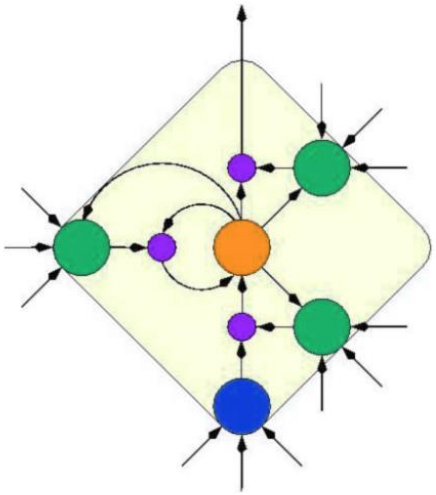
Three different classes of network architectures

- single-layer feed-forward
- multi-layer feed-forward
- recurrent

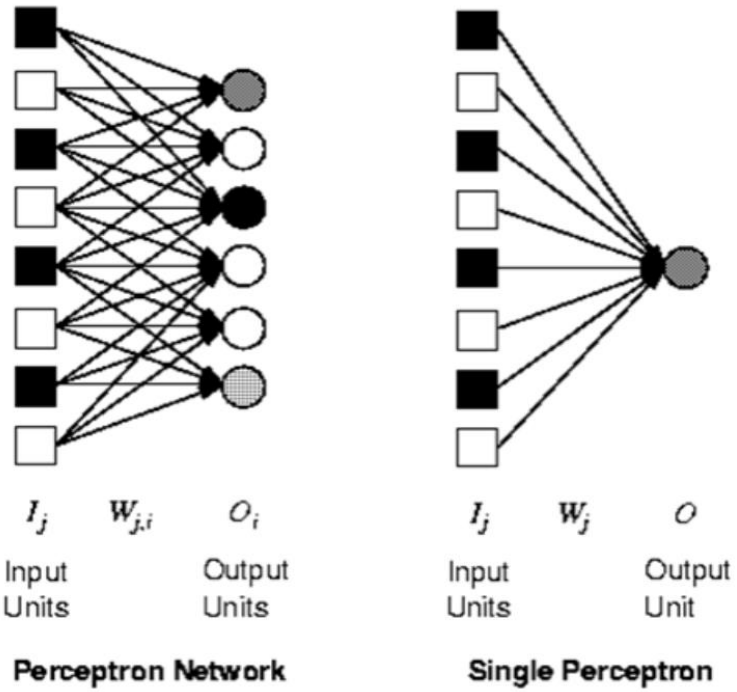
} neurons are organized in acyclic layers







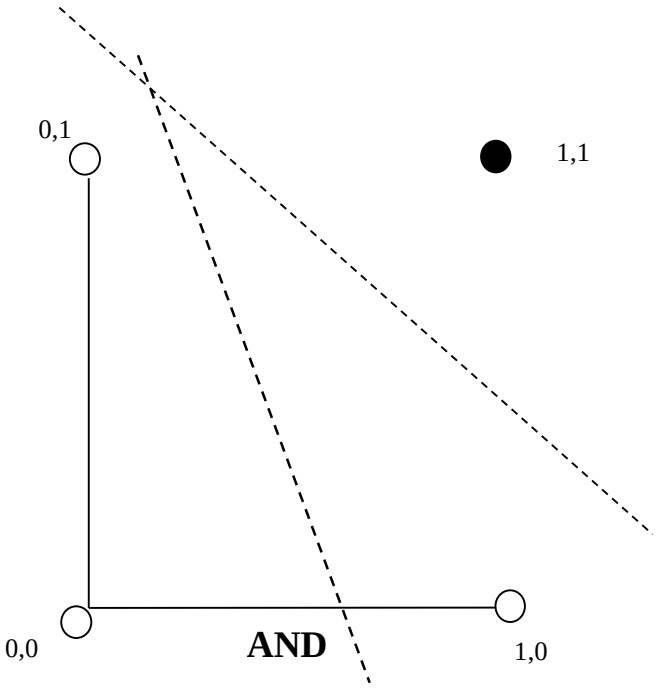
PERCEPTRON



Synonym for Single-Layer, Feed-Forward Network

- First Studied in the 1950's

LINEAR SEPARABILITY

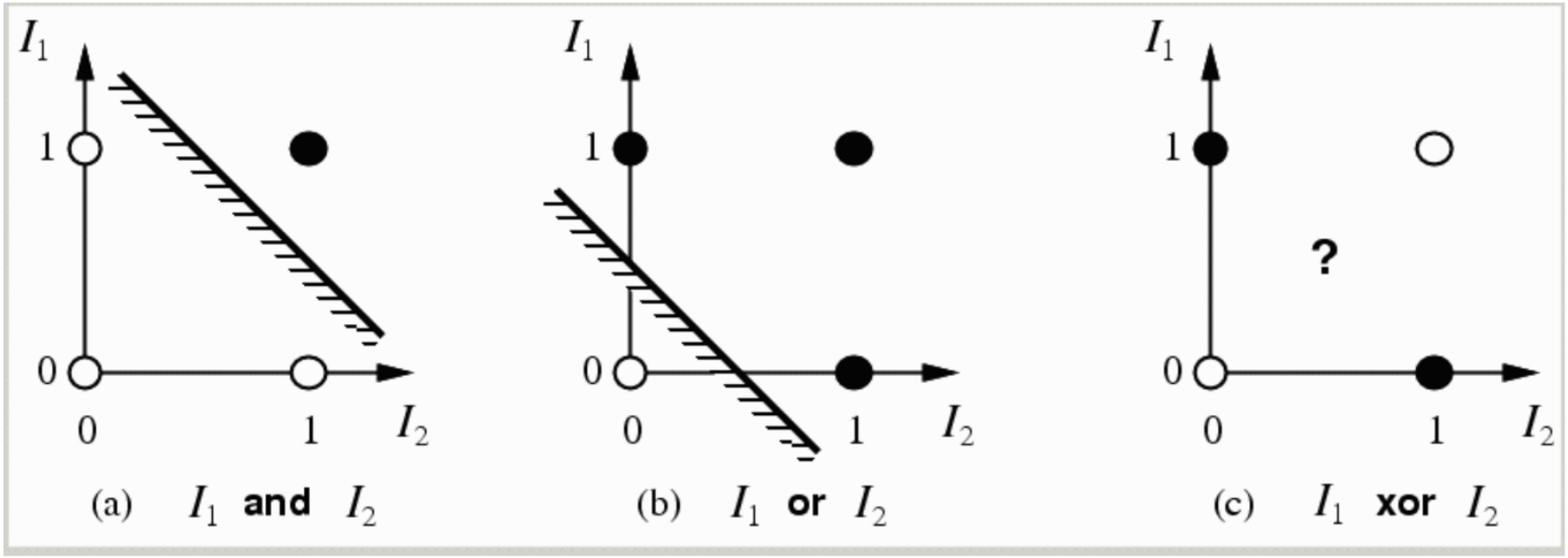


	AND			
Input 1	0	0	1	1
Input 2	0	1	0	1
Output	0	0	0	1

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

LINEAR SEPARABILITY

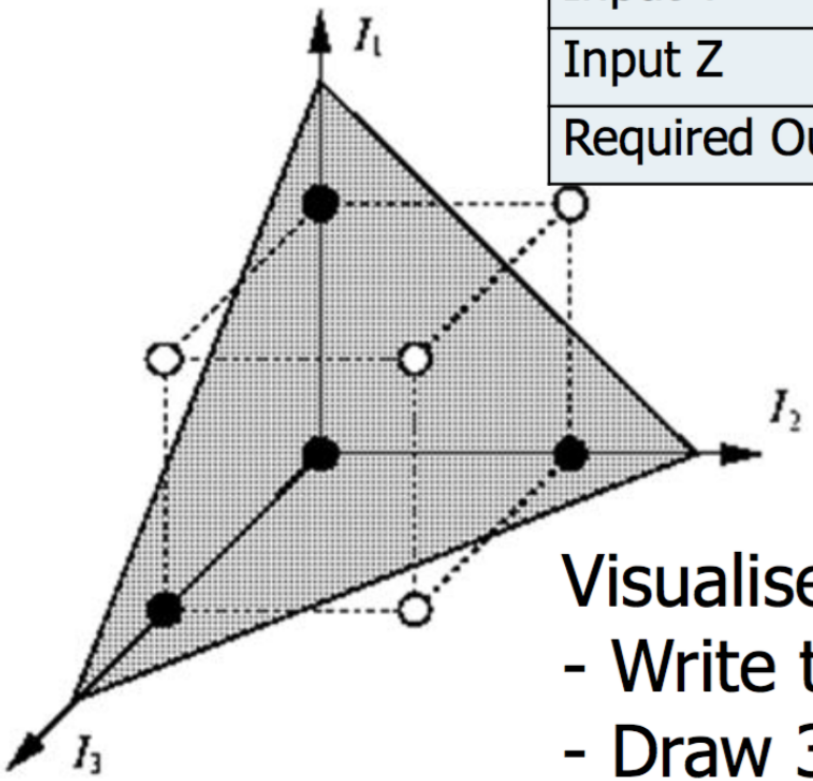
Minsky & Papert



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

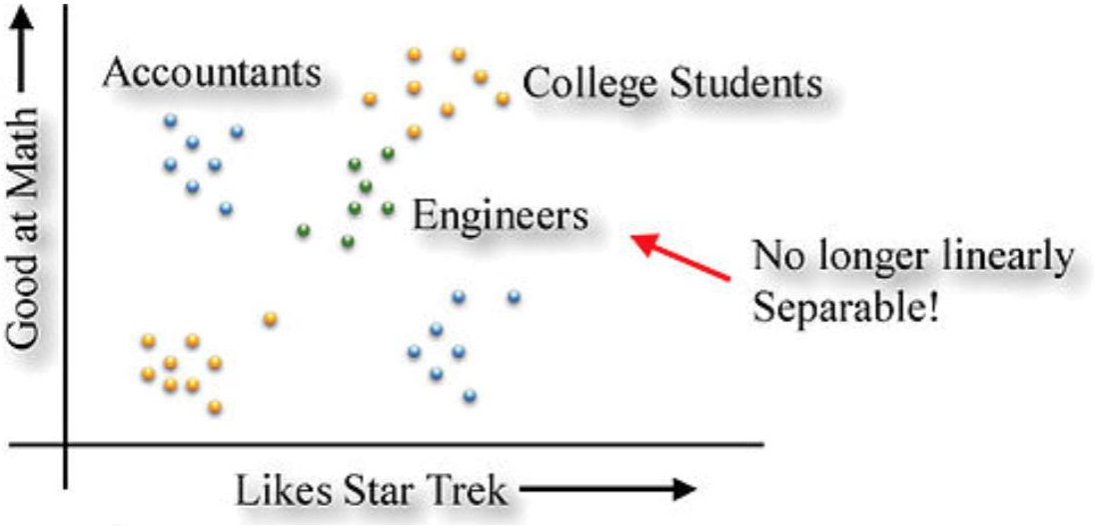
LINEAR SEPARABILITY

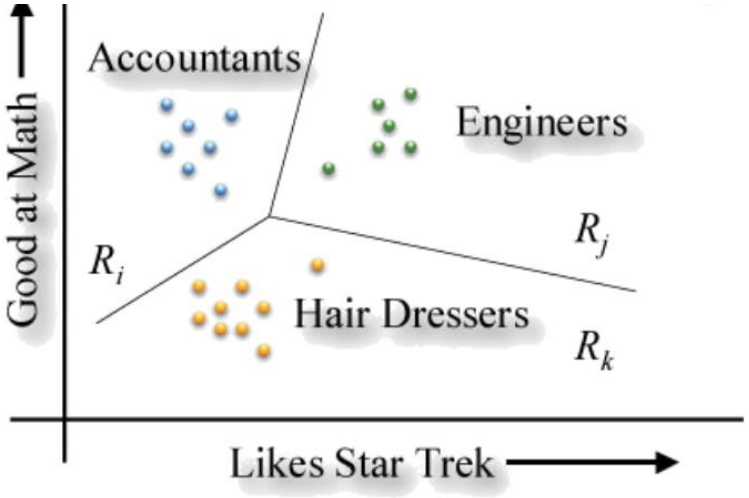
Input X	0	0	1	1	0	0	1	1
Input Y	0	1	0	1	0	1	0	1
Input Z	0	0	0	0	1	1	1	1
Required Output	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**0**



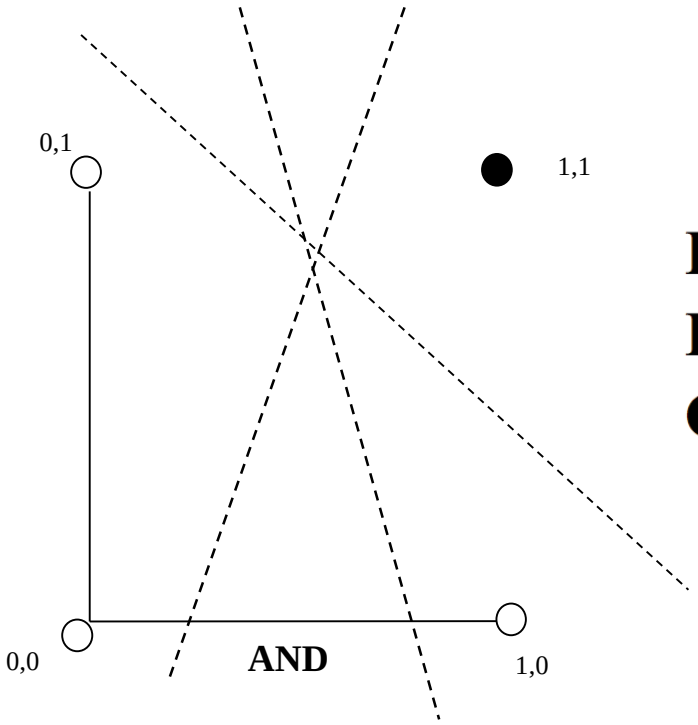
Visualise ANN with 3 inputs
- Write the truth table
- Draw 3D graphical representation

LINEAR SEPARABILITY

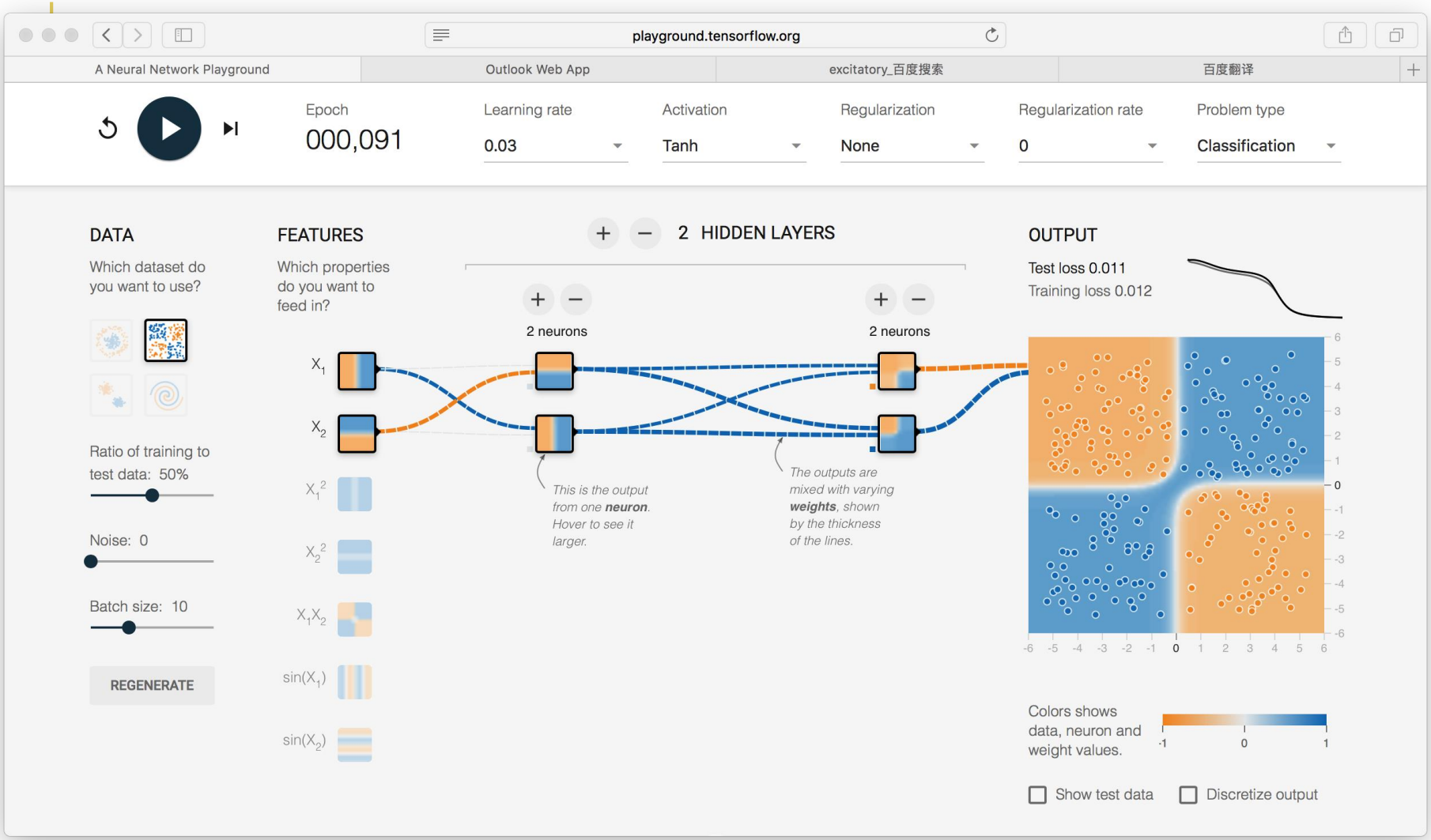




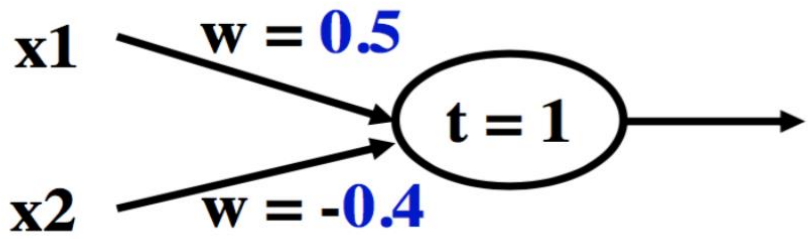
TRAINING A NN



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


playground.tensorflow.org
A Neural Network Playground
Outlook Web App
excitatory_百度搜索
百度翻译
Epoch
000,091
Learning rate
0.03
Activation
Tanh
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
2 neurons
2 neurons
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.011
Training loss 0.012
Colors shows data, neuron and weight values.
Show test data
Discretize output

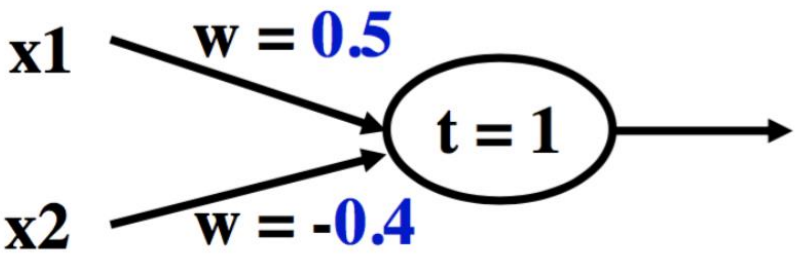
Training a NN



AND		
X1	X2	T
0	0	0
0	1	0
1	0	0
1	1	1

X1	X2	Summation	Output
0	0	(0 * 0.5) + (0 * -0.4) = 0	0
0	1		
1	0		
1	1		

Training a NN



AND			
X1	X2	T	O
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	0

X1	X2	Summation	Output
0	0	(0 * 0.5) + (0 * -0.4) = 0	0
0	1	(0 * 0.5) + (1 * -0.4) = -0.4	0
1	0	(1 * 0.5) + (0 * -0.4) = 0.5	0
1	1	(1 * 0.5) + (1 * -0.4) = 0.1	0

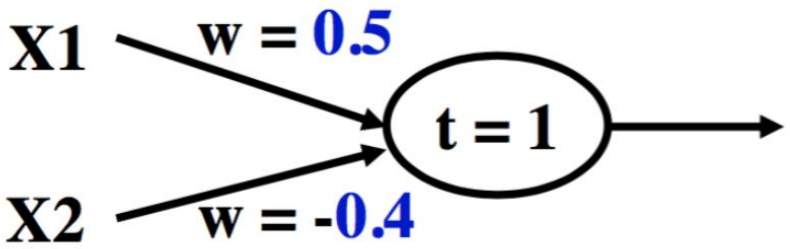
Training a NN

AND			
X1	X2	T	O
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	0

While epoch produces an error

Check next inputs (pattern) from epoch

End While



Epoch: The entire training set feed into the neural network. The AND function: an epoch consists of four sets of inputs (patterns) feed into the network ([0,0], [0,1], [1,0], [1,1])

Training a NN

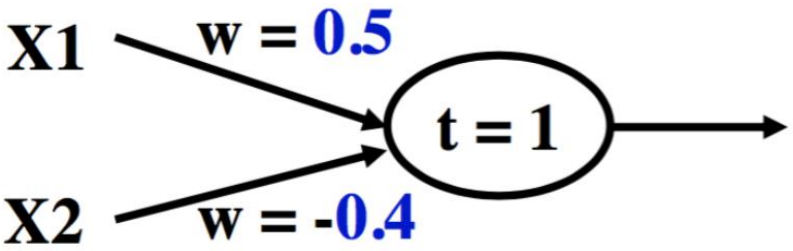
AND			
X1	X2	T	O
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	0

While epoch produces an error

Check next inputs (pattern) from epoch

Err = T – O

End While



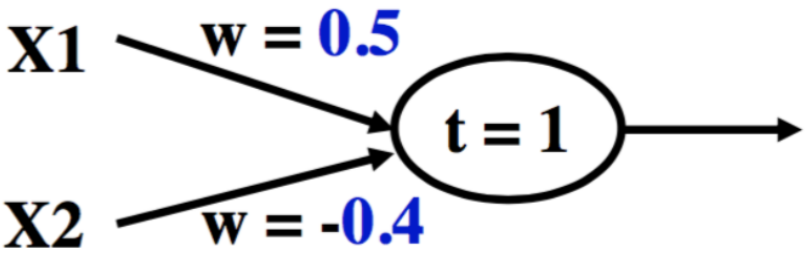
Training Value, T : The value that we require the network to produce. For example, with [0,0] for the AND function the training value will be 0.

Training a NN

AND			
X1	X2	T	O
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	0

While epoch produces an error
  Check next inputs (pattern) from epoch
  Err = T – O
  If Err <> 0 then

  End If
End While



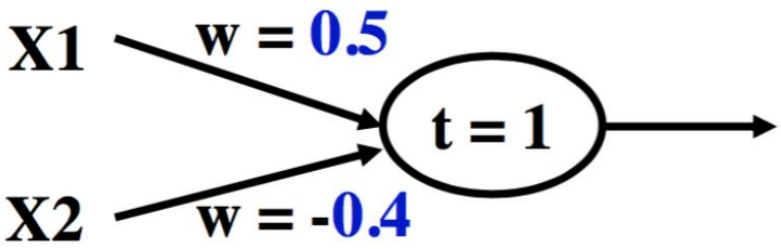
Error, Err : The amount the output by the network O differs from the training value T.

Training a NN

What is the problem if the learning rate is set too high, or too low?

AND			
X1	X2	T	O
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	0

While epoch produces an error
  Check next inputs (pattern) from epoch
  Err = T – O
  If Err <> 0 then
    $w_i = w_i + LR * X_i * Err$
  End If
End While



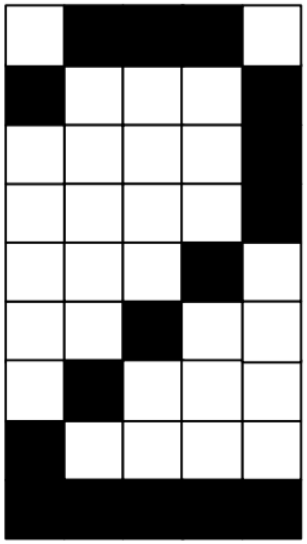
X_i: Inputs to the neuron

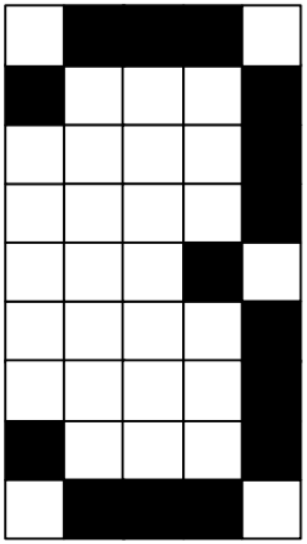
w_i: Weight from input X_i to the output

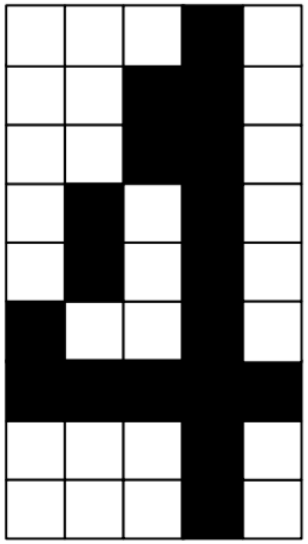
LR: The learning rate: how quickly the network converges. It is set by experimentation, typically 0.1

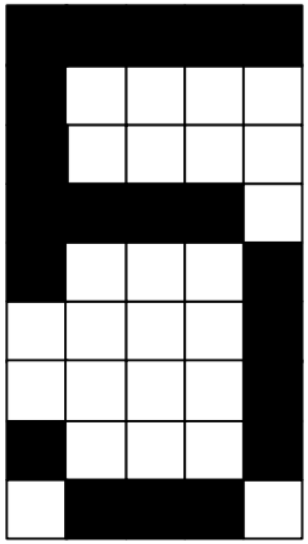
Bit Maps for Digit Recognition

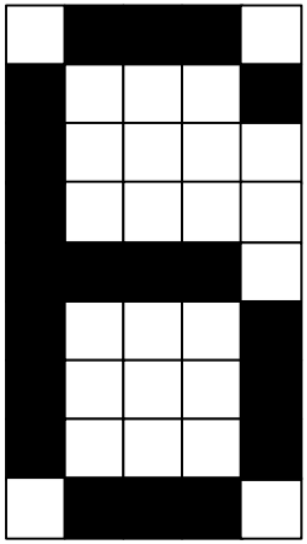
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30
31	32	33	34	35
36	37	38	39	40
41	42	43	44	45

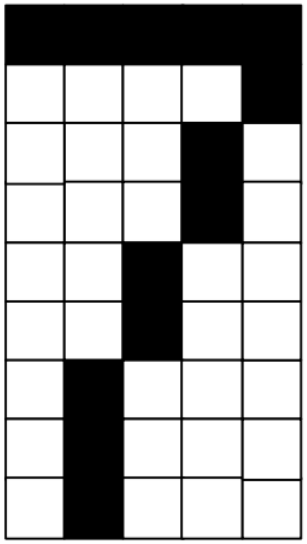


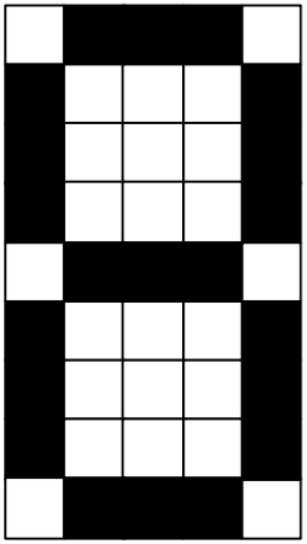


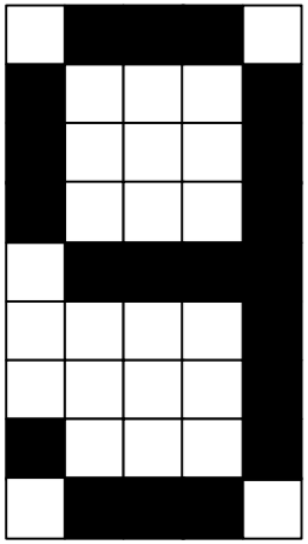


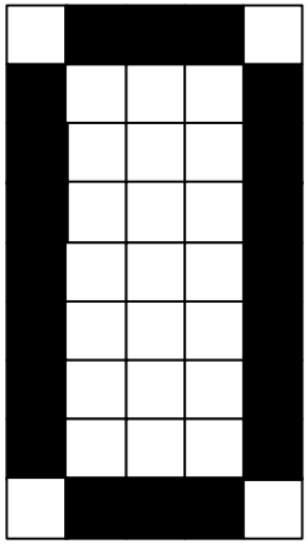




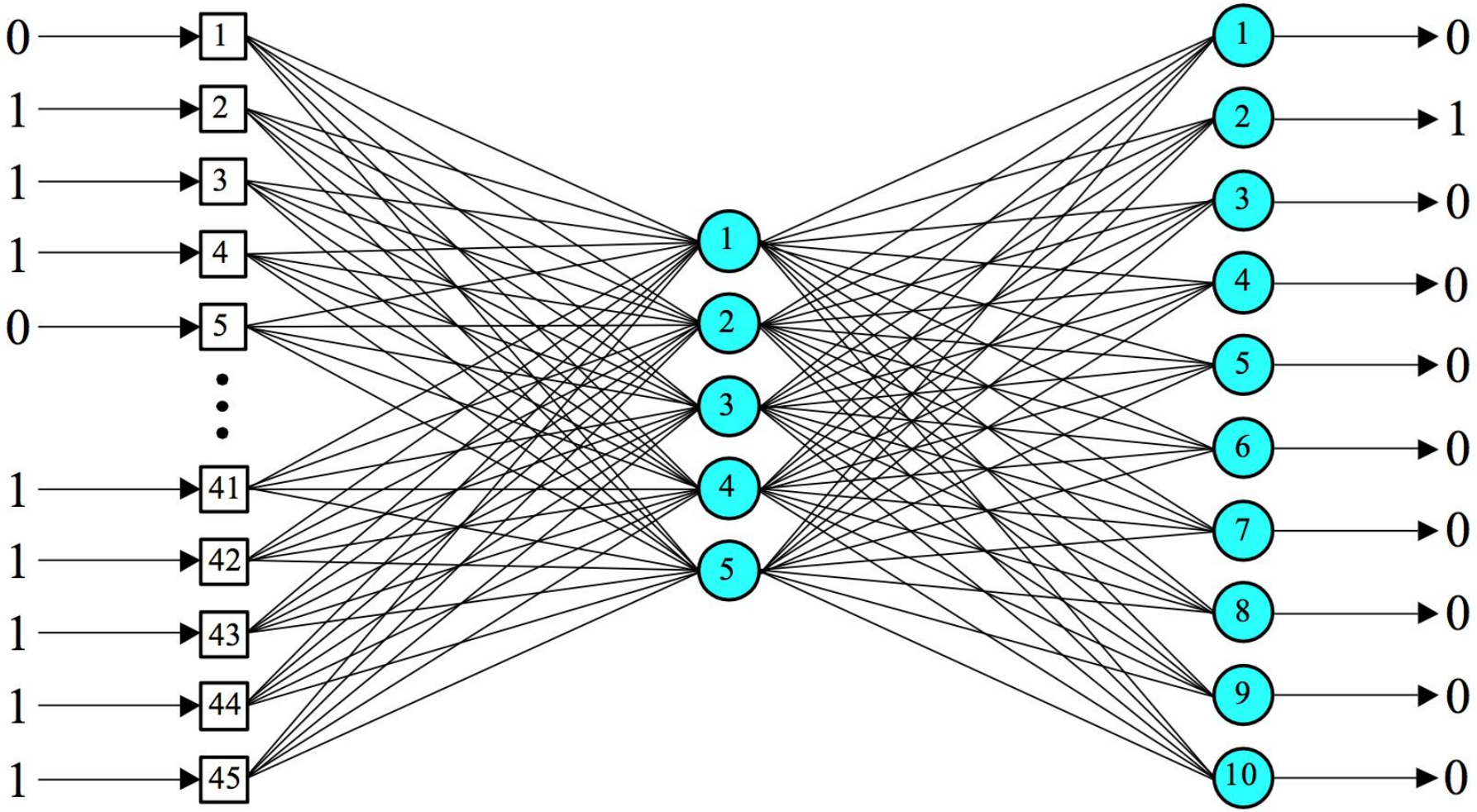






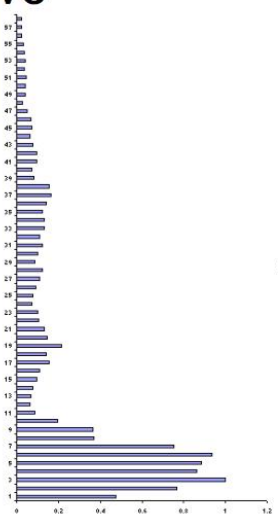


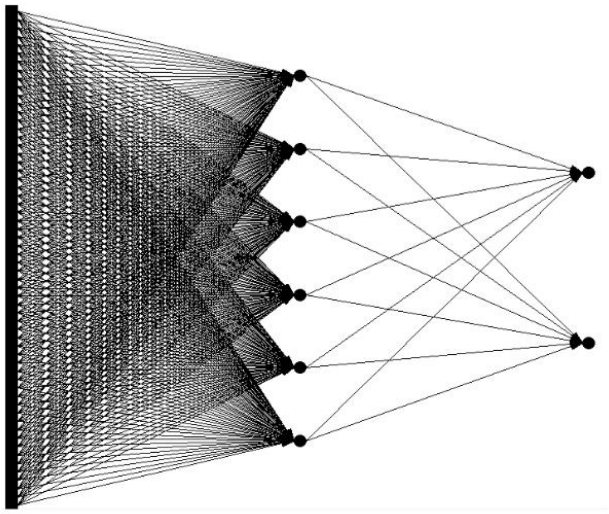
Neural Network for Printed Digit Recognition



Voice Recognition

Steve

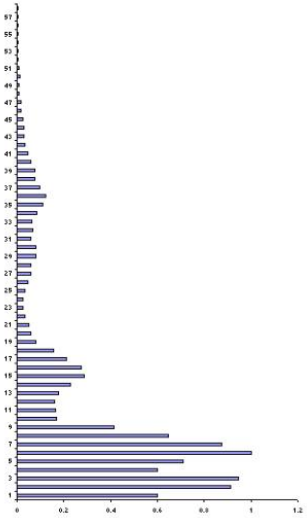


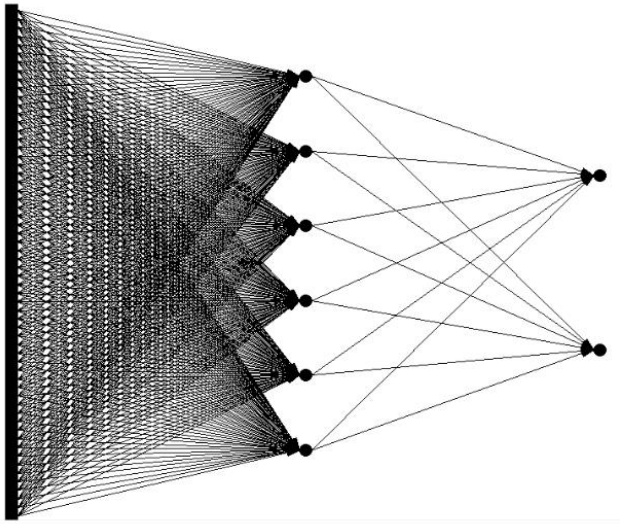


0.01

0.99

David





0.99

0.01

Network Architecture

Feedforward Network

60 inputs (one for each frequency bin);

6 hidden;

2 output (0-1 for "Steve", 1-0 for "David")

SUMMARY

First neural networks (McCulloch-Pitts)

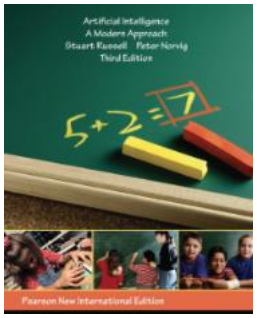
Simple networks

Perceptron (single layer NN)

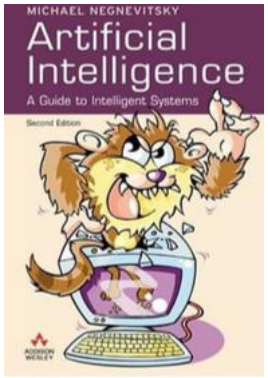
- Representation
- Limitations (linearly separable)
- Learning

FURTHER READING

AIMA Chapter 1.2.4, 18.7



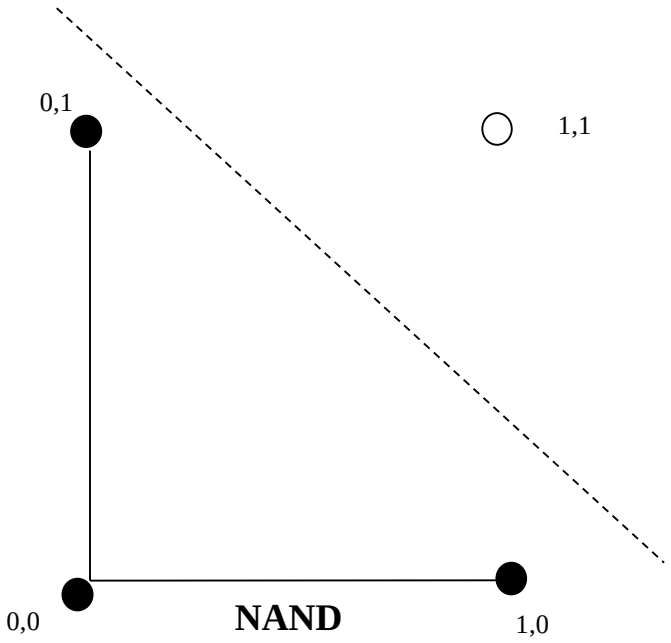
AIIS Chapter 6



TRAINING A PERCEPTRON

Can we train a perceptron to learn logical NAND?

	NAND			
Input 1	**0**	**0**	**1**	**1**
Input 2	**0**	**1**	**0**	**1**
Output	**1**	**1**	**1**	**0**



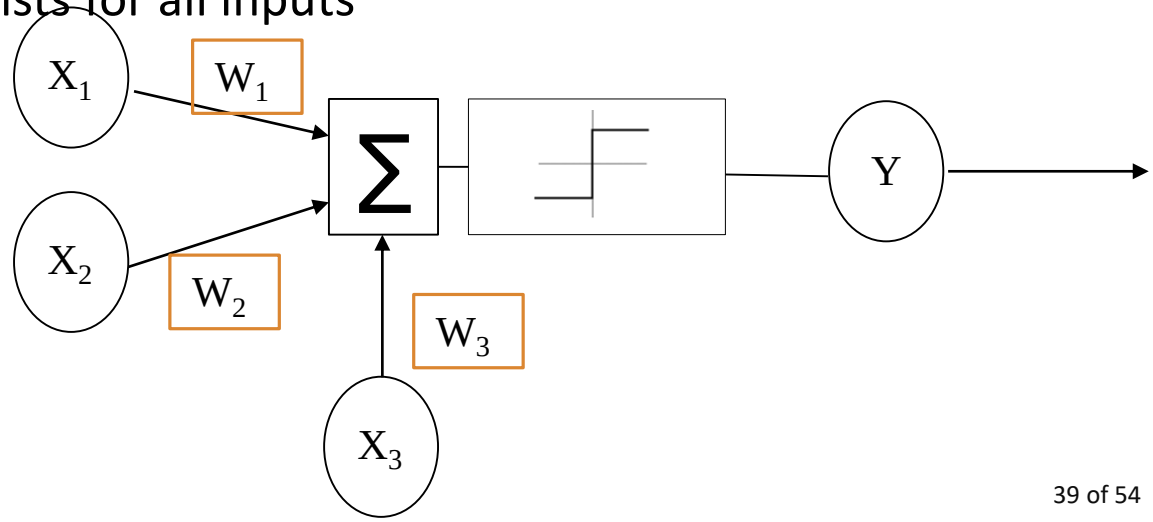
THE LEARNING PROCESS

Randomly assign initial weights

Do:

- Present the network with input
- Calculate the error value in the output
- Adjust the weightings of the inputs according to the error

Until no error value exists for all inputs

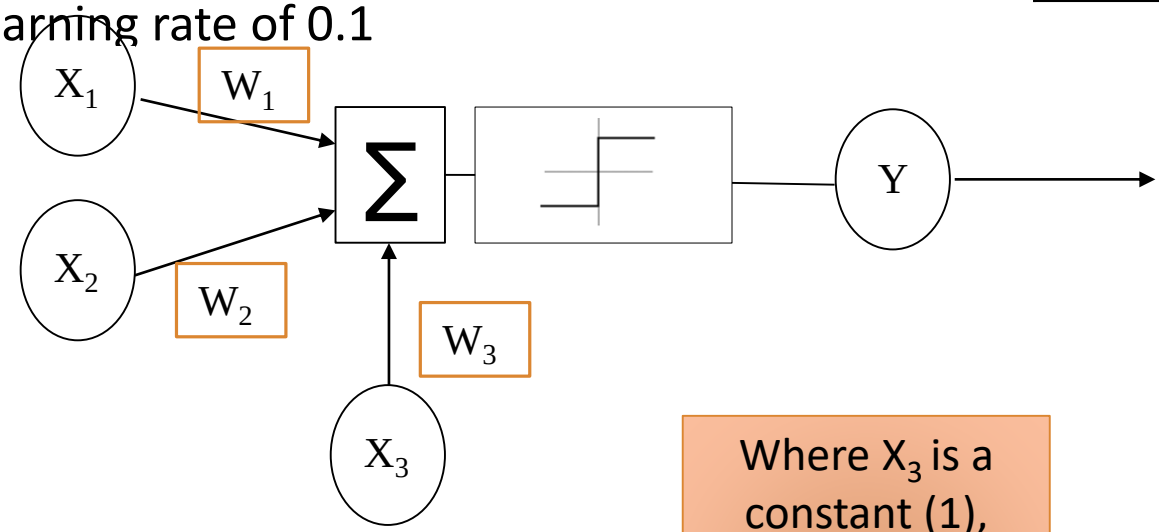


TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

Given a threshold value of 0.6

And a learning rate of 0.1

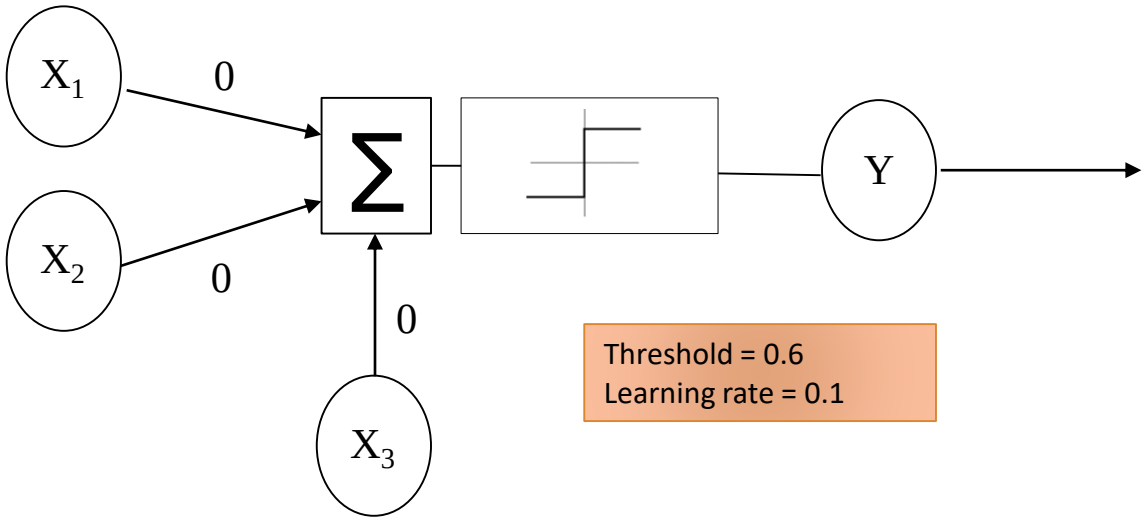


Where X_3 is a constant (1), weighting the perceptron within the network

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.0	0.0	0.0			
0	1	1							
1	0	1							
1	1	1							

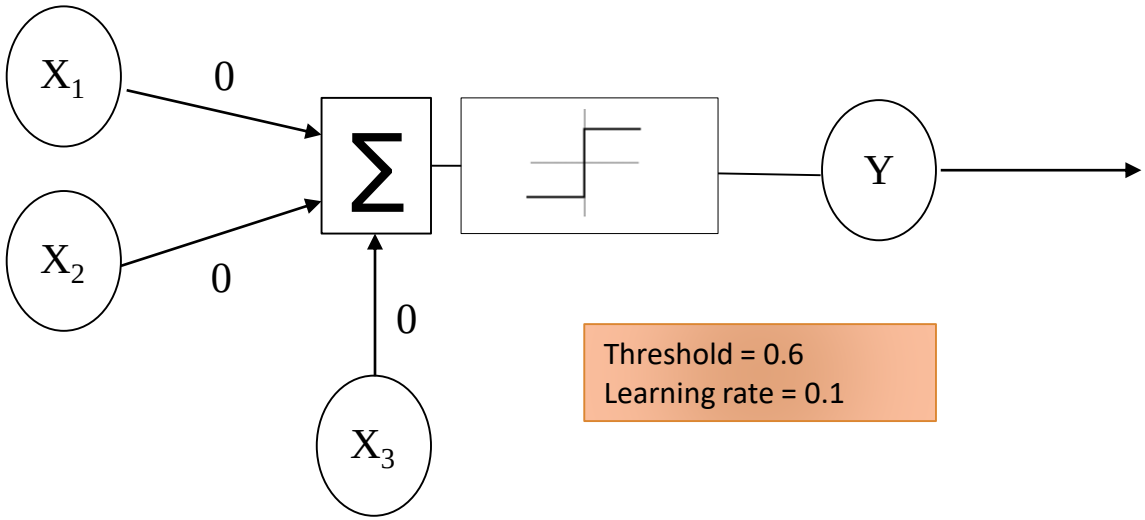


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.0	0.0	0.0	0	1	+0.1
0	1	1							
1	0	1							
1	1	1							

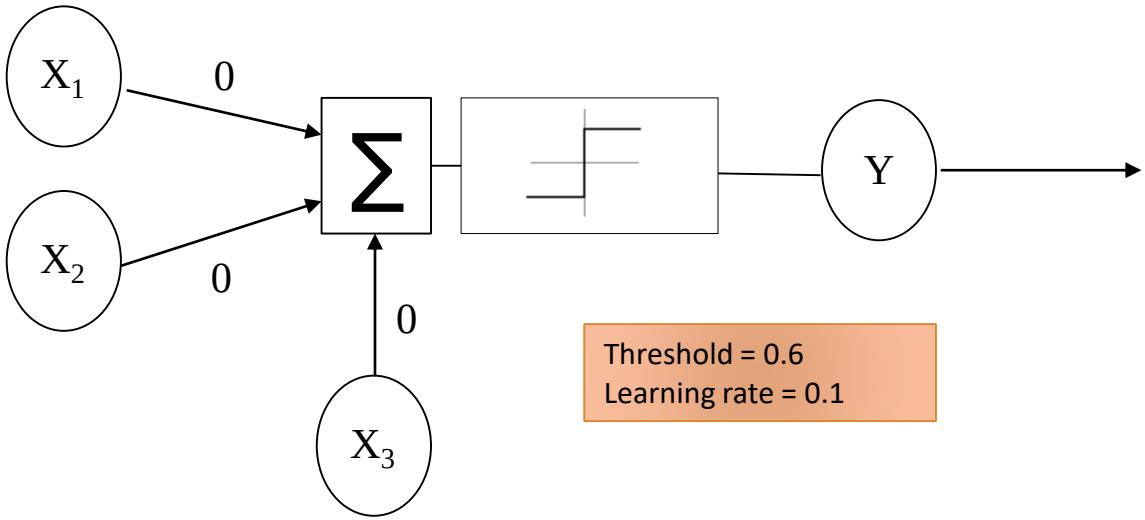


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.0	0.0	0.0	0	1	+0.1
0	1	1	0.0	0.0	0.1	0.1			
1	0	1							
1	1	1							

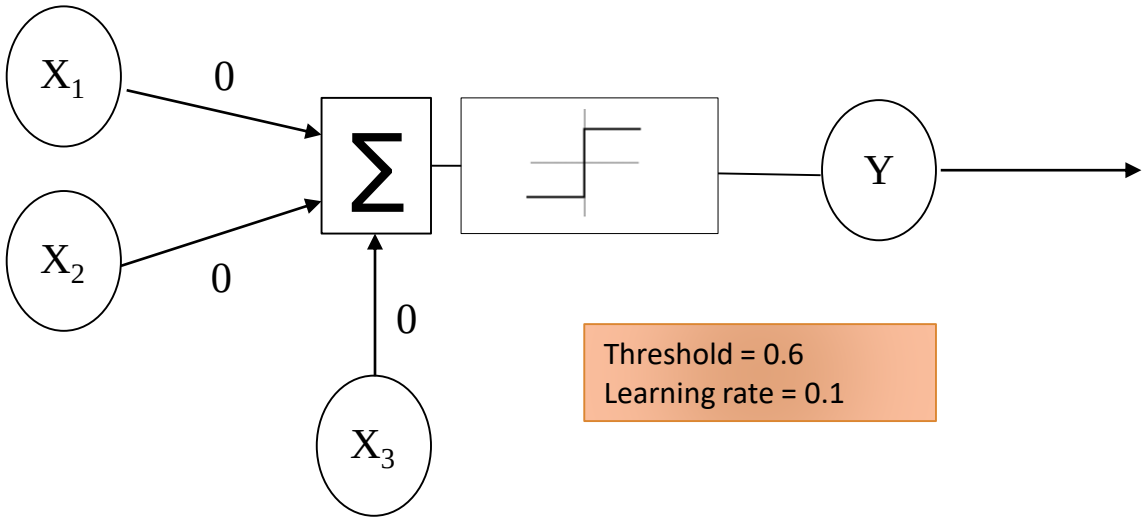


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.0	0.0	0.0	0	1	+0.1
0	1	1	0.0	0.0	0.1	0.1	0	1	+0.1
1	0	1							
1	1	1							

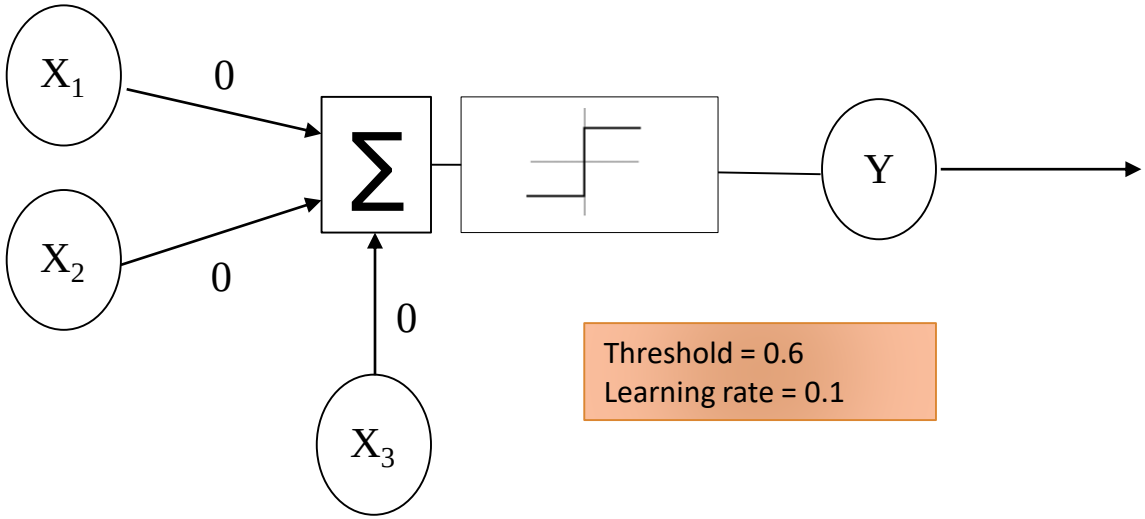


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.0	0.0	0.0	0	1	+0.1
0	1	1	0.0	0.0	0.1	0.1	0	1	+0.1
1	0	1	0.0	0.1	0.2				
1	1	1							

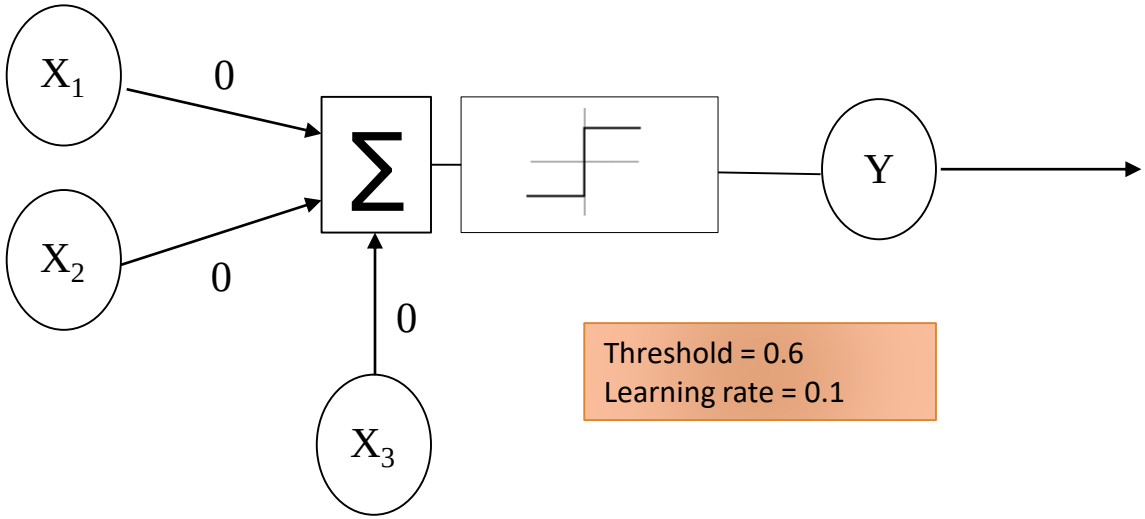


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.0	0.0	0.0	0	1	+0.1
0	1	1	0.0	0.0	0.1	0.1	0	1	+0.1
1	0	1	0.0	0.1	0.2	0.2	0	1	+0.1
1	1	1							

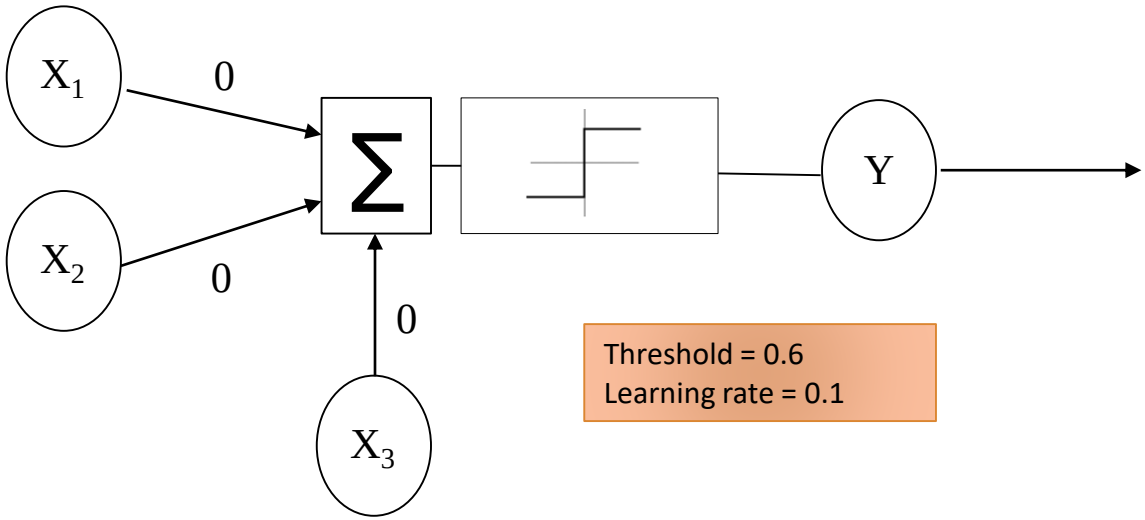


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.0	0.0	0.0	0	1	+0.1
0	1	1	0.0	0.0	0.1	0.1	0	1	+0.1
1	0	1	0.0	0.1	0.2	0.2	0	1	+0.1
1	1	1	0.1	0.1	0.3	0.5			

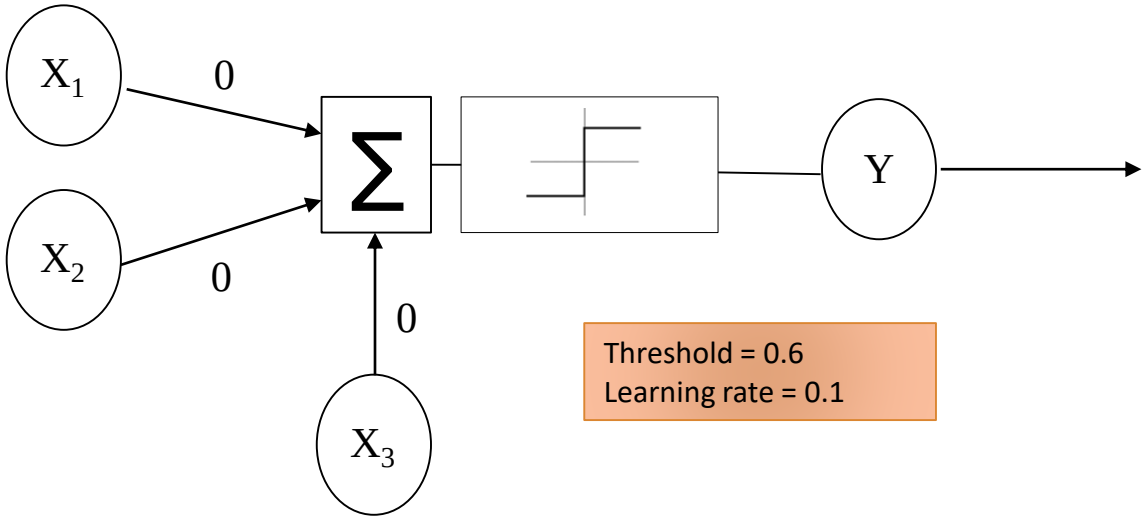


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.0	0.0	0.0	0	1	+0.1
0	1	1	0.0	0.0	0.1	0.1	0	1	+0.1
1	0	1	0.0	0.1	0.2	0.2	0	1	+0.1
1	1	1	0.1	0.1	0.3	0.5	0	0	∓0.0

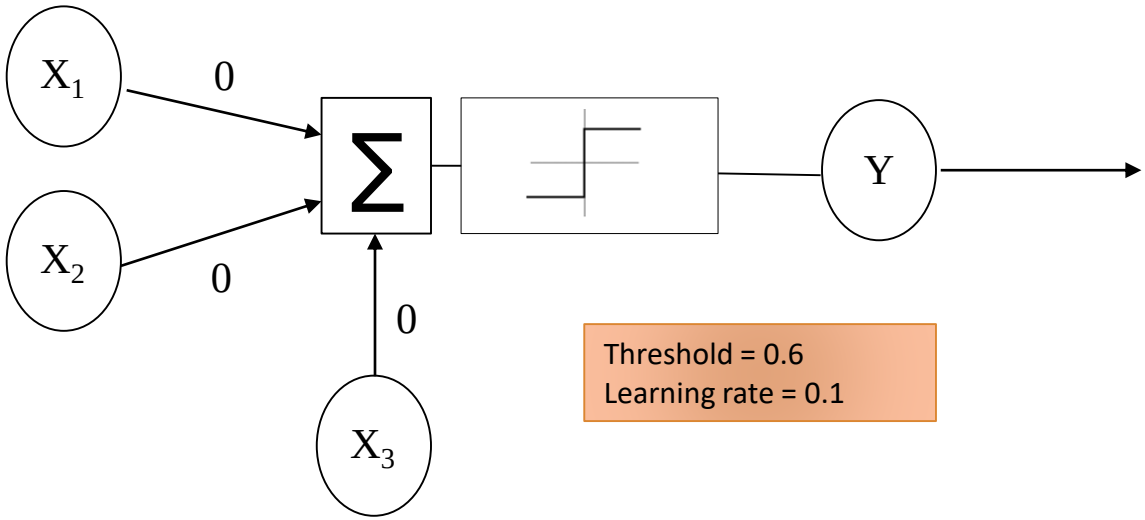


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.1	0.1	0.3	0.3			
0	1	1							
1	0	1							
1	1	1							

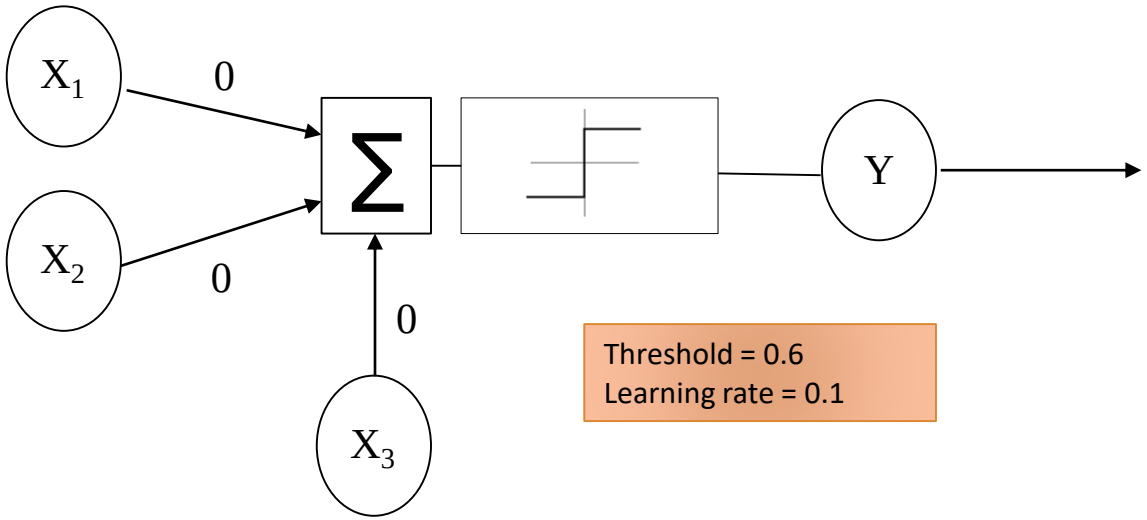


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.1	0.1	0.3	0.3	0	1	+0.1
0	1	1							
1	0	1							
1	1	1							

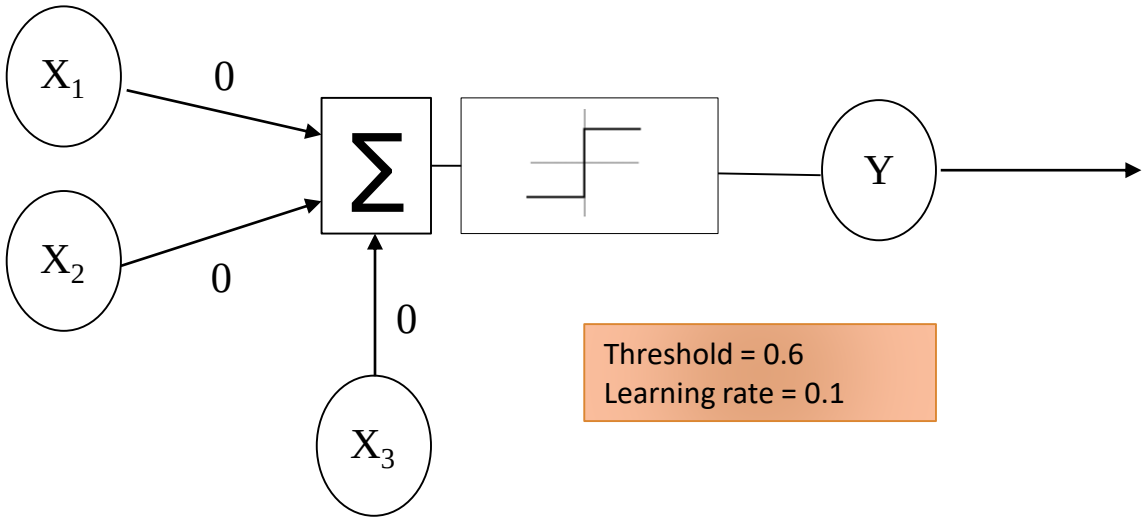


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.1	0.1	0.3	0.3	0	1	+0.1
0	1	1	0.1	0.1	0.4	0.5			
1	0	1							
1	1	1							

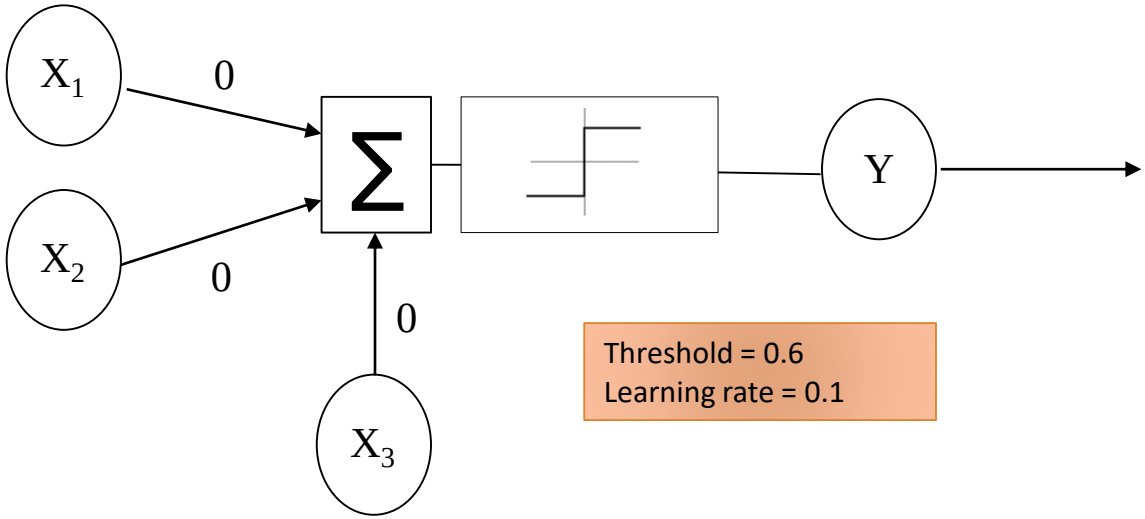


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.1	0.1	0.3	0.3	0	1	+0.1
0	1	1	0.1	0.1	0.4	0.5	0	1	+0.1
1	0	1							
1	1	1							

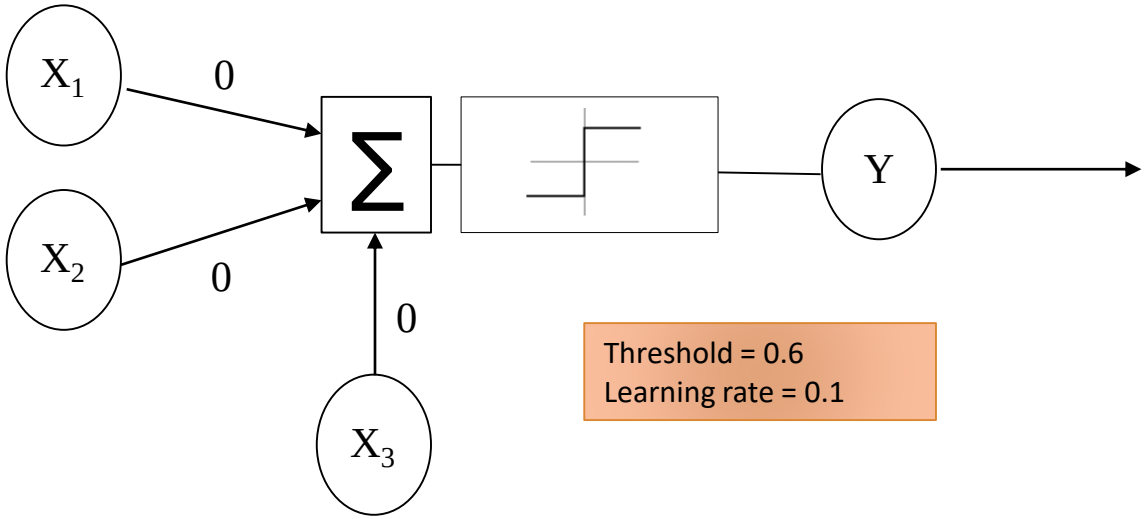


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.1	0.1	0.3	0.3	0	1	+0.1
0	1	1	0.1	0.1	0.4	0.5	0	1	+0.1
1	0	1	0.1	0.2	0.5	0.6			
1	1	1							

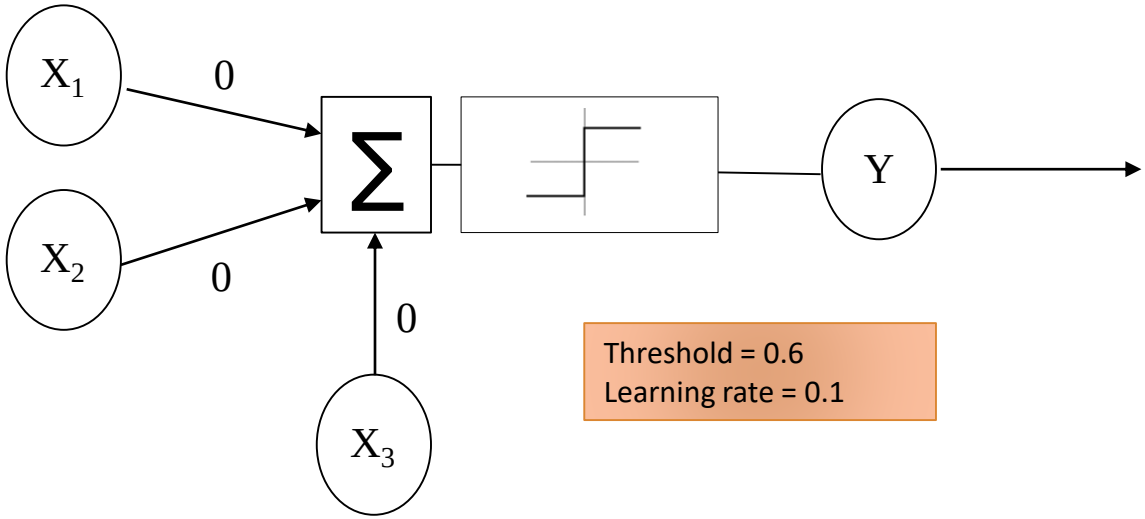


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.1	0.1	0.3	0.3	0	1	+0.1
0	1	1	0.1	0.1	0.4	0.5	0	1	+0.1
1	0	1	0.1	0.2	0.5	0.6	1	0	∓0.0
1	1	1							

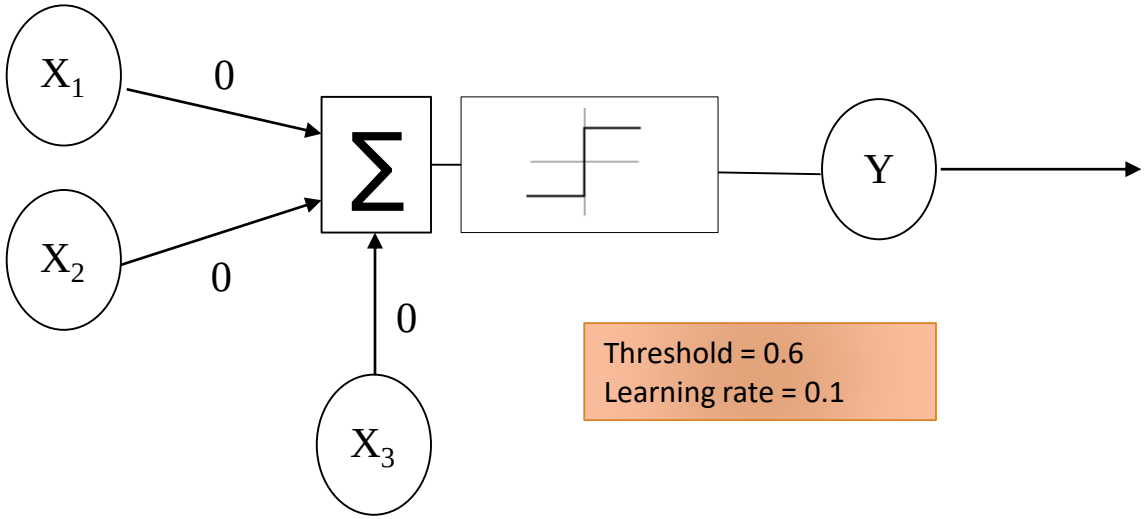


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.1	0.1	0.3	0.3	0	1	+0.1
0	1	1	0.1	0.1	0.4	0.5	0	1	+0.1
1	0	1	0.1	0.2	0.5	0.6	1	0	∓0.0
1	1	1	0.1	0.2	0.5	0.8			

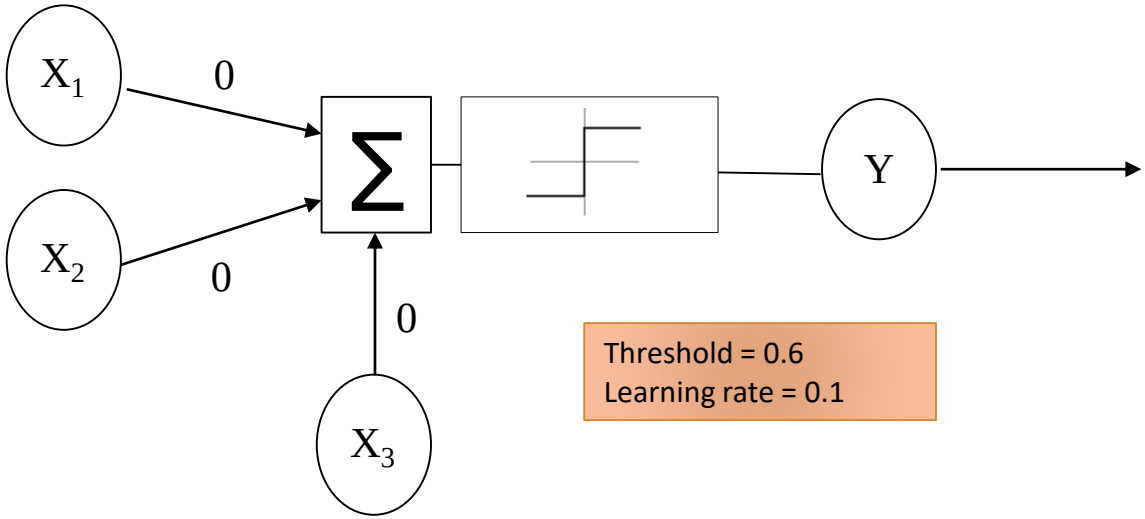


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.1	0.1	0.3	0.3	0	1	+0.1
0	1	1	0.1	0.1	0.4	0.5	0	1	+0.1
1	0	1	0.1	0.2	0.5	0.6	1	0	∓0.0
1	1	1	0.1	0.2	0.5	0.8	1	-1	-0.1

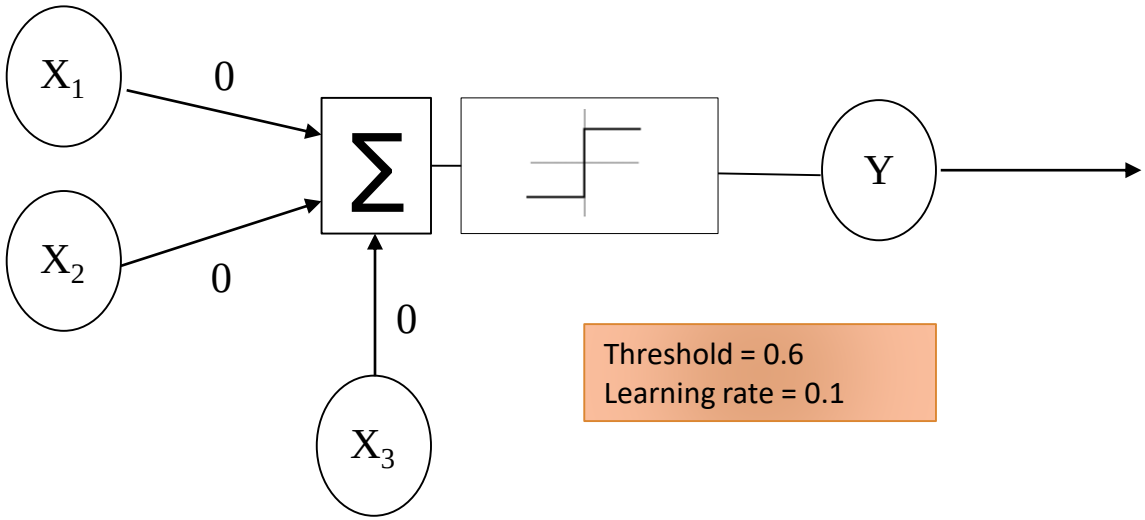


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.1	0.4				
0	1	1							
1	0	1							
1	1	1							

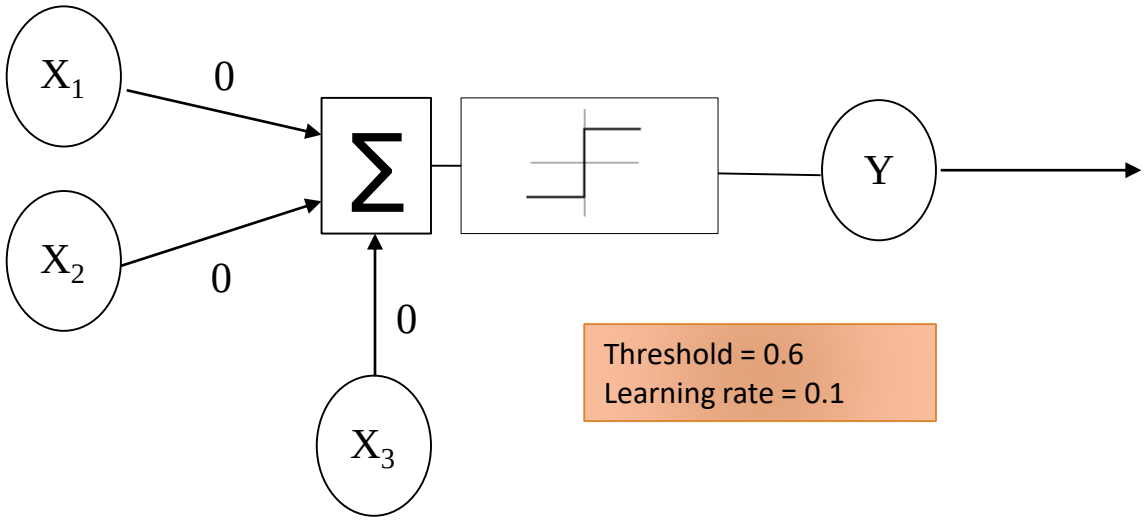


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.1	0.4	0.4	0	1	+0.1
0	1	1							
1	0	1							
1	1	1							

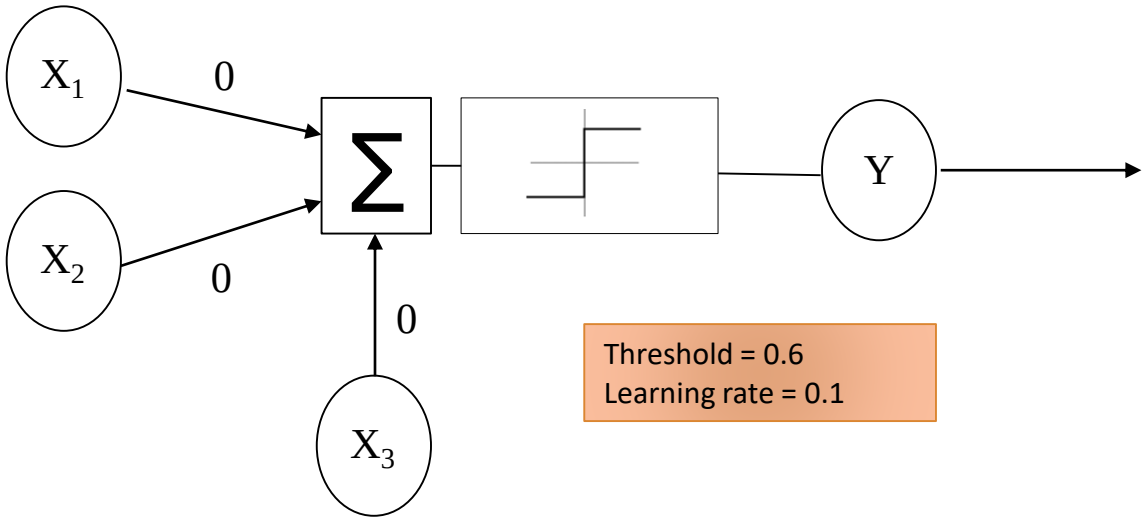


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.1	0.4	0.4	0	1	+0.1
0	1	1	0.0	0.1	0.5				
1	0	1							
1	1	1							

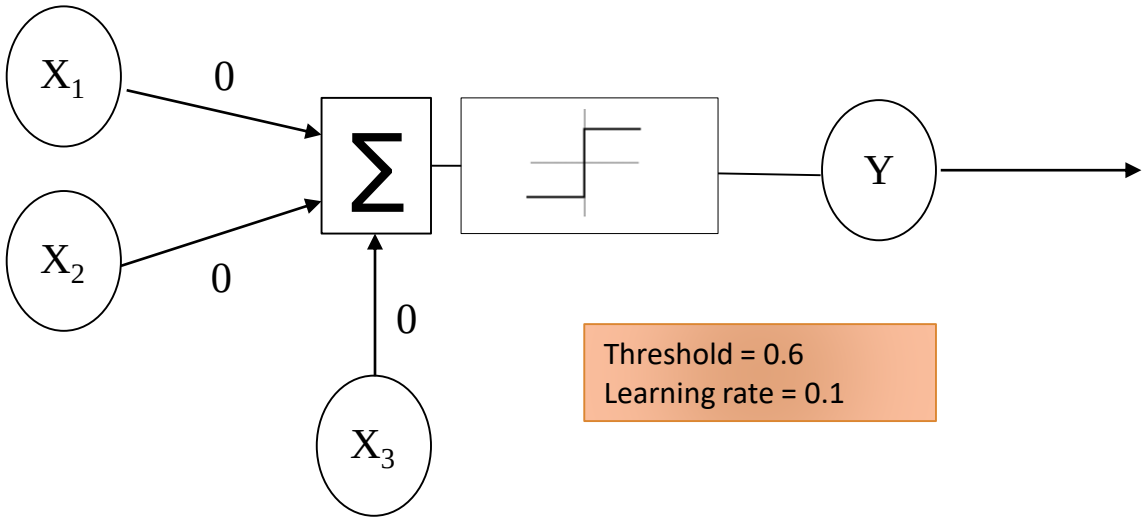


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.1	0.4	0.4	0	1	+0.1
0	1	1	0.0	0.1	0.5	0.6	1	0	∓0.0
1	0	1							
1	1	1							

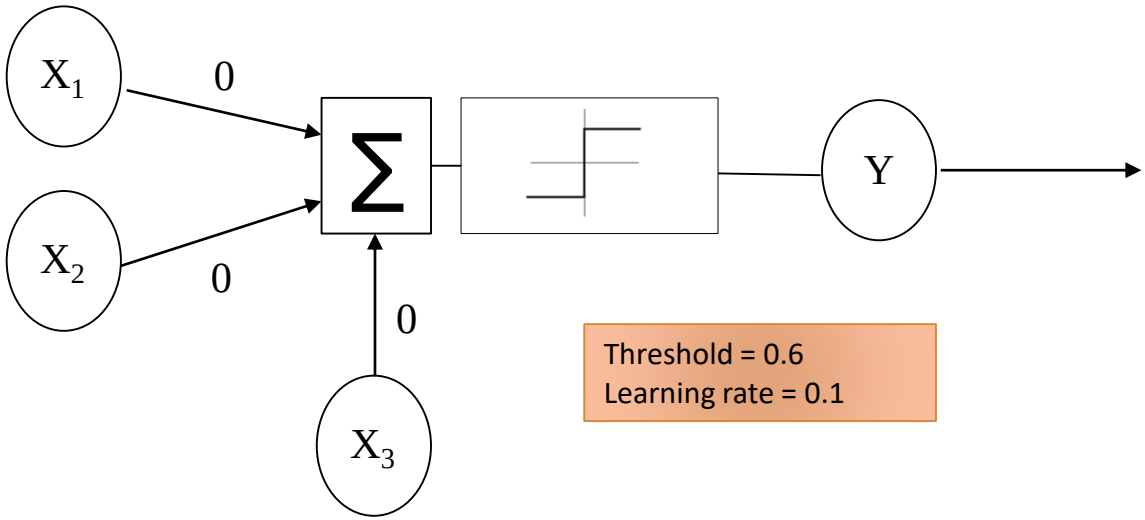


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.1	0.4	0.4	0	1	+0.1
0	1	1	0.0	0.1	0.5	0.6	1	0	∓0.0
1	0	1	0.0	0.1	0.5	0.5	0	1	+0.1
1	1	1							

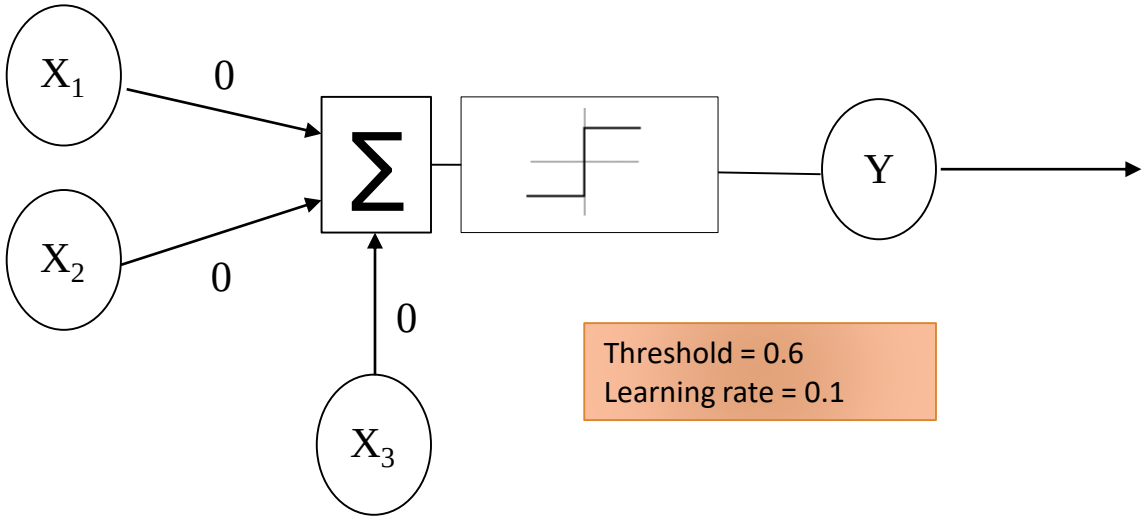


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

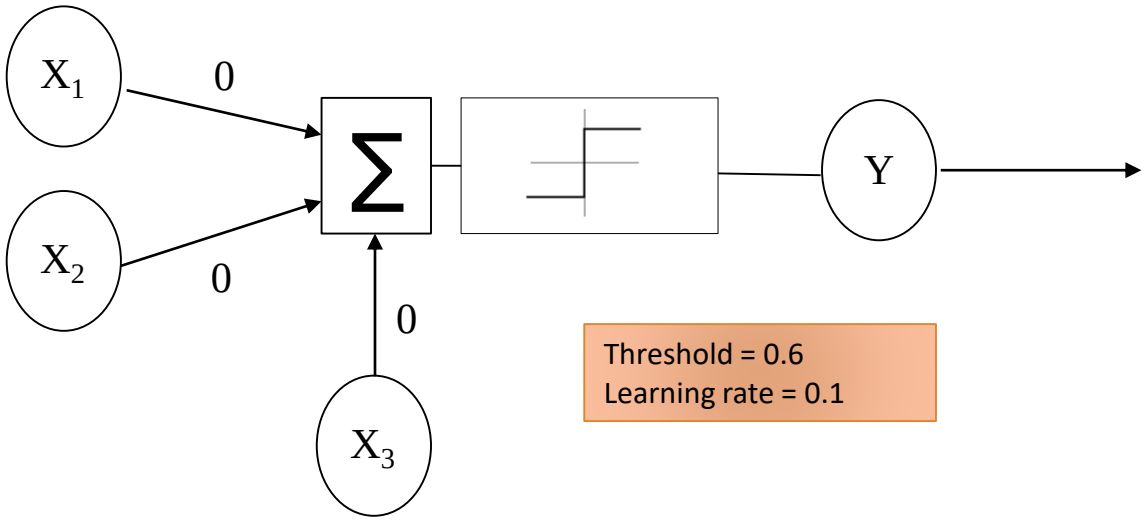
X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.1	0.4	0.4	0	1	+0.1
0	1	1	0.0	0.1	0.5	0.6	1	0	∓0.0
1	0	1	0.0	0.1	0.5	0.5	0	1	+0.1
1	1	1	0.1	0.1	0.6	0.8	1	-1	-0.1



TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.0	0.5	0.5	0	1	+0.1
0	1	1							
1	0	1							
1	1	1							

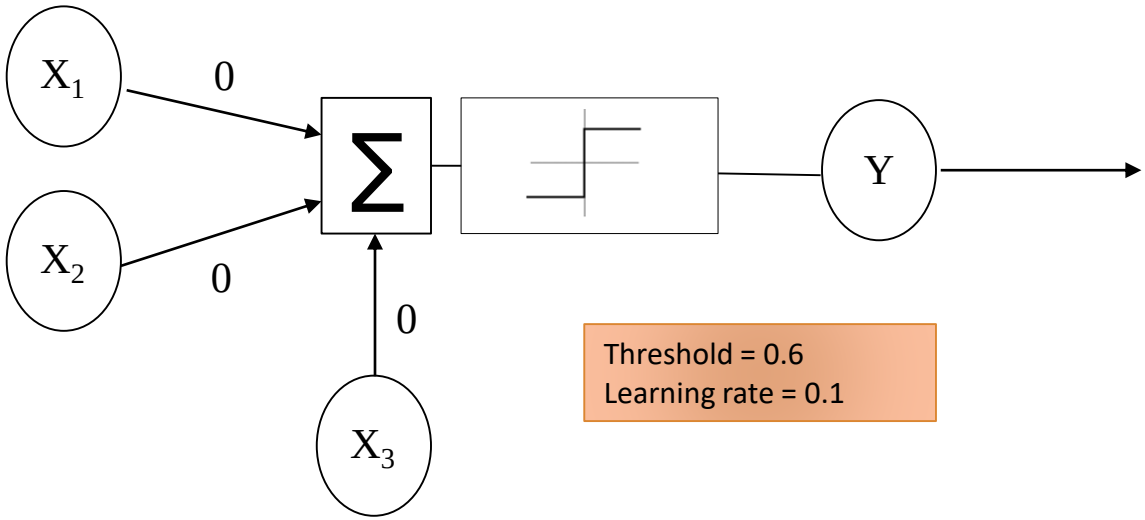


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.0	0.5	0.5	0	1	+0.1
0	1	1	0.0	0.0	0.6	0.6	1	0	∓0.0
1	0	1							
1	1	1							

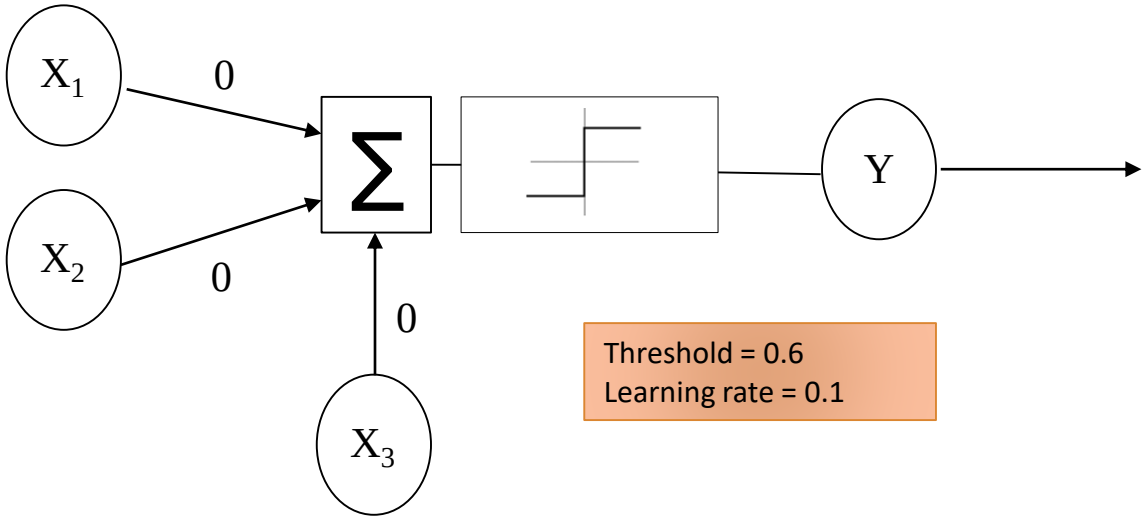


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.0	0.5	0.5	0	1	+0.1
0	1	1	0.0	0.0	0.6	0.6	1	0	∓0.0
1	0	1	0.0	0.0	0.6	0.6	1	0	∓0.0
1	1	1							

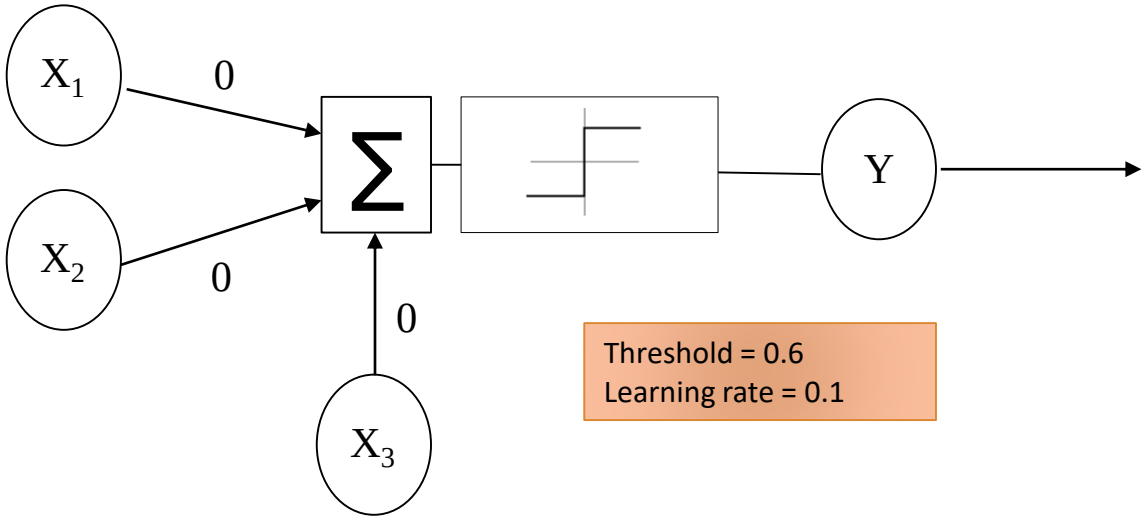


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.0	0.5	0.5	0	1	+0.1
0	1	1	0.0	0.0	0.6	0.6	1	0	∓0.0
1	0	1	0.0	0.0	0.6	0.6	1	0	∓0.0
1	1	1	0.0	0.0	0.6	0.6	1	-1	-0.1

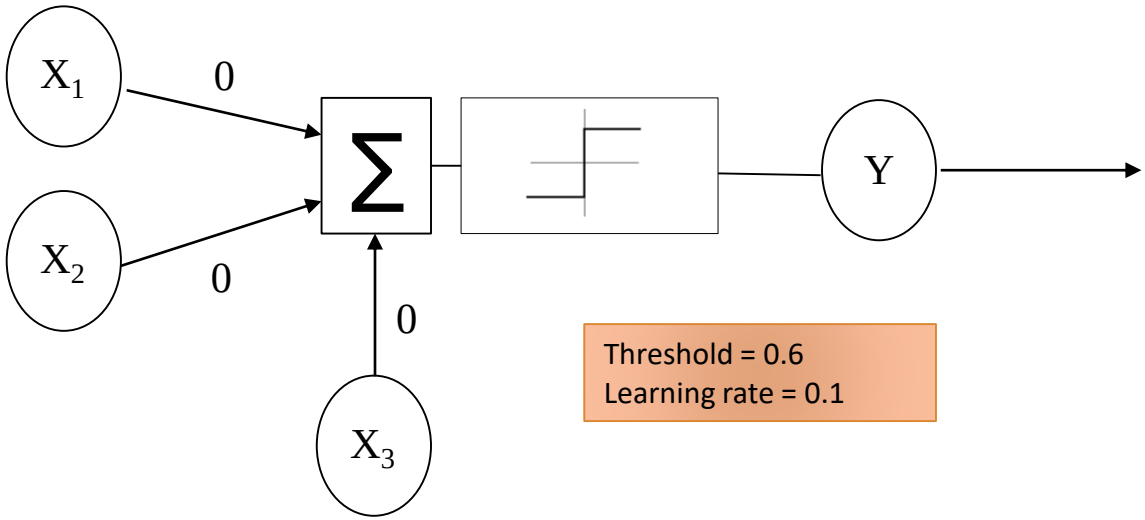


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.1	-0.1	0.5	0.5	0	1	+0.1
0	1	1							
1	0	1							
1	1	1							

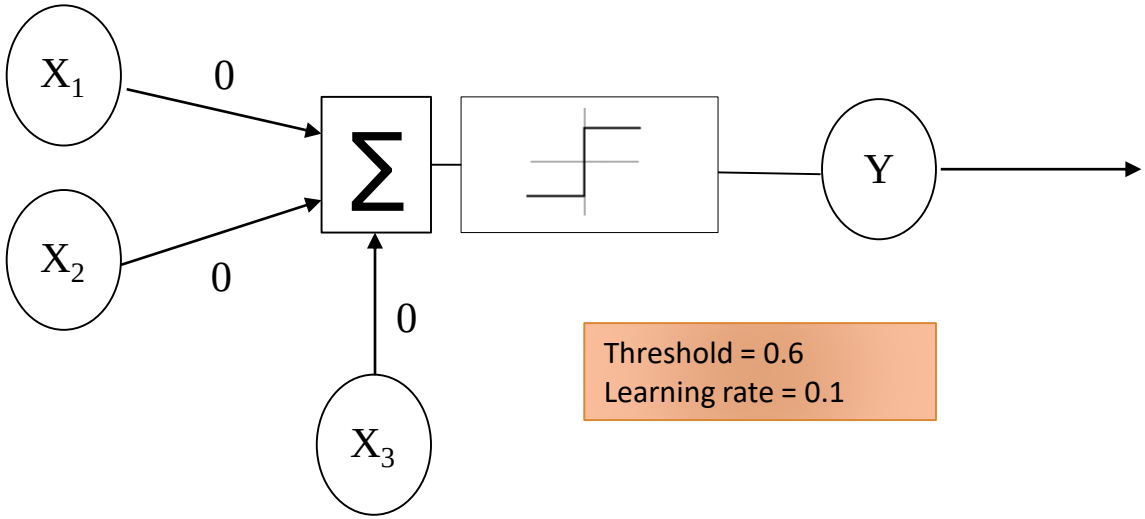


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.1	-0.1	0.5	0.5	0	1	+0.1
0	1	1	-0.1	-0.1	0.6	0.5	0	1	+0.1
1	0	1							
1	1	1							

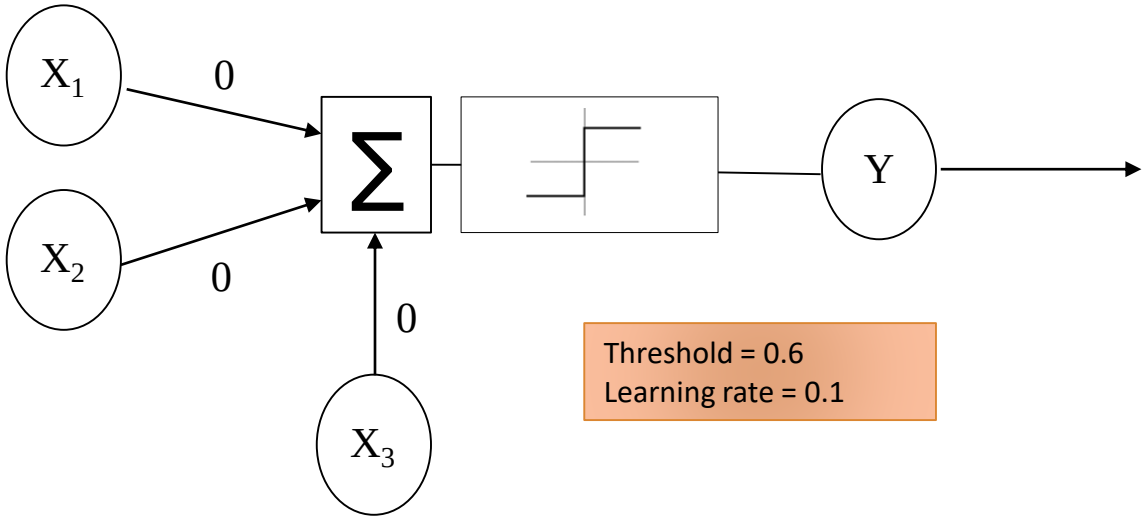


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.1	-0.1	0.5	0.5	0	1	+0.1
0	1	1	-0.1	-0.1	0.6	0.5	0	1	+0.1
1	0	1	-0.1	0.0	0.7	0.6	1	0	∓0.0
1	1	1							

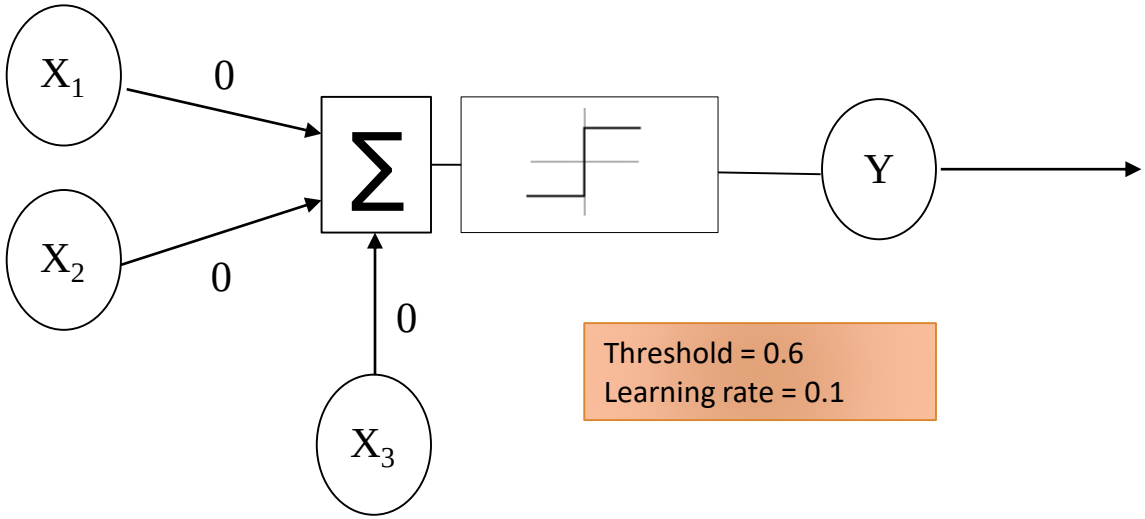


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.1	-0.1	0.5	0.5	0	1	+0.1
0	1	1	-0.1	-0.1	0.6	0.5	0	1	+0.1
1	0	1	-0.1	0.0	0.7	0.6	1	0	∓0.0
1	1	1	-0.1	0.0	0.7	0.6	1	-1	-0.1

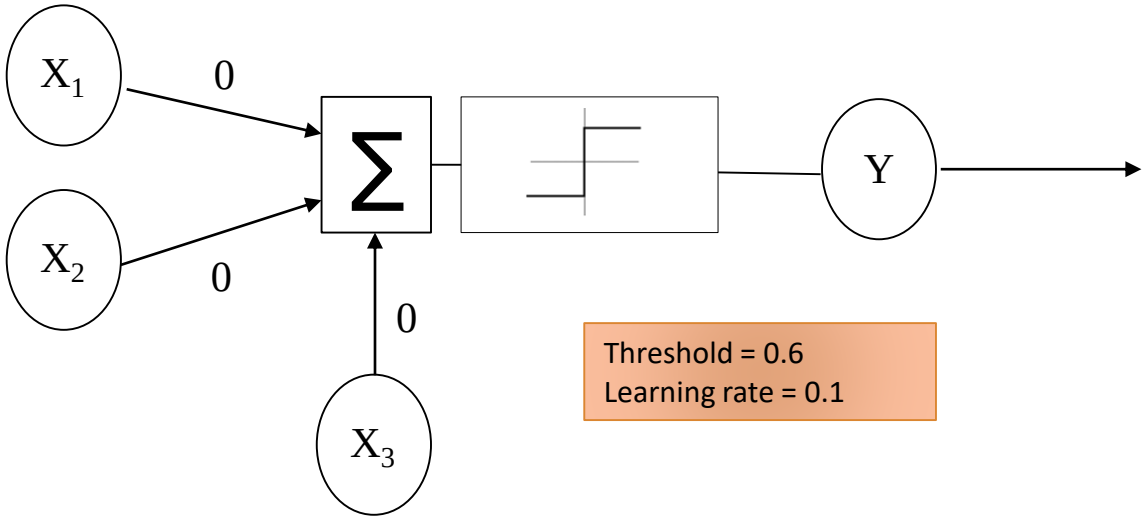


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

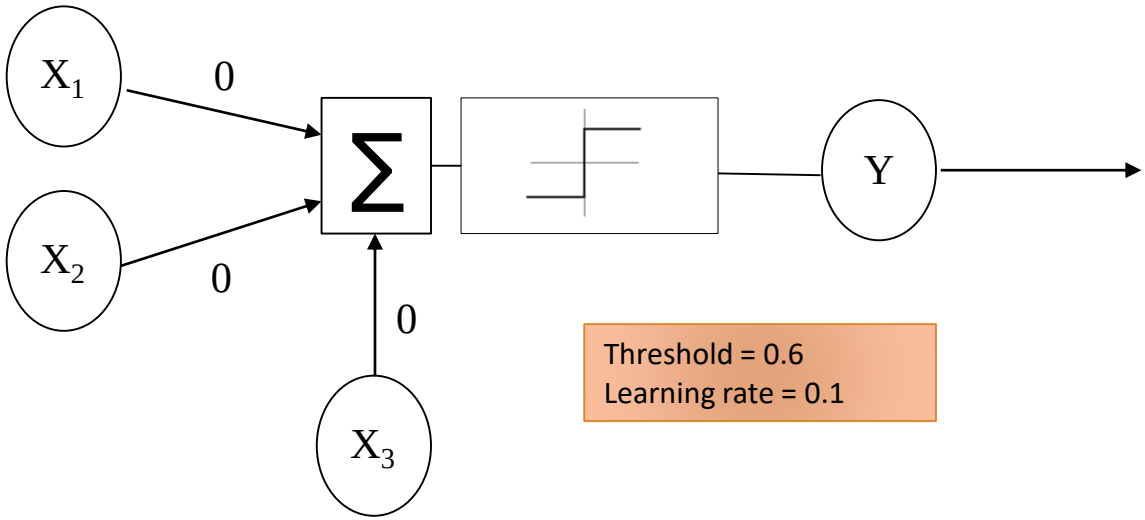
X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.2	-0.1	0.6	0.6	1	0	$\mp$0.0
0	1	1	-0.2	-0.1	0.6	0.5	0	1	+0.1
1	0	1	-0.2	0.0	0.7	0.5	0	1	+0.1
1	1	1	-0.1	0.0	0.8	0.7	1	-1	-0.1



TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.2	-0.1	0.7				
0	1	1							
1	0	1							
1	1	1							

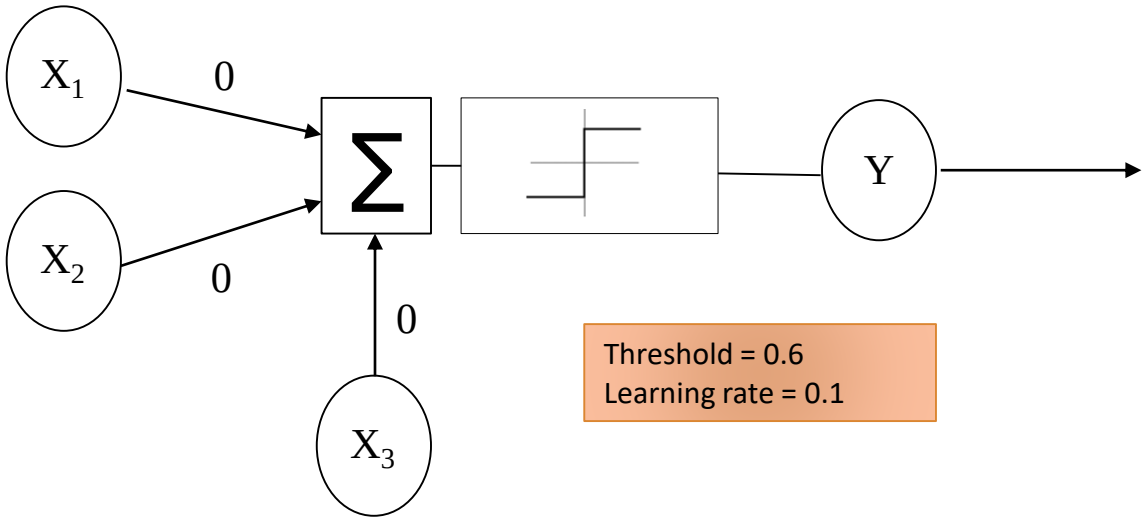


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.2	-0.1	0.7	0.7	1	0	$\mp$0.0
0	1	1	-0.2	-0.1	0.7	0.6	1	0	$\mp$0.0
1	0	1							
1	1	1							

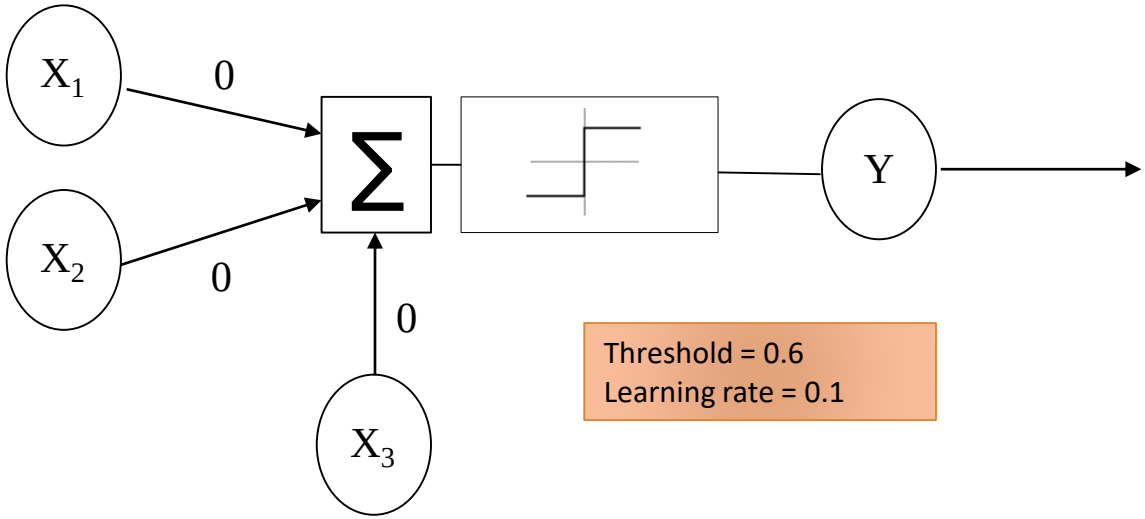


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

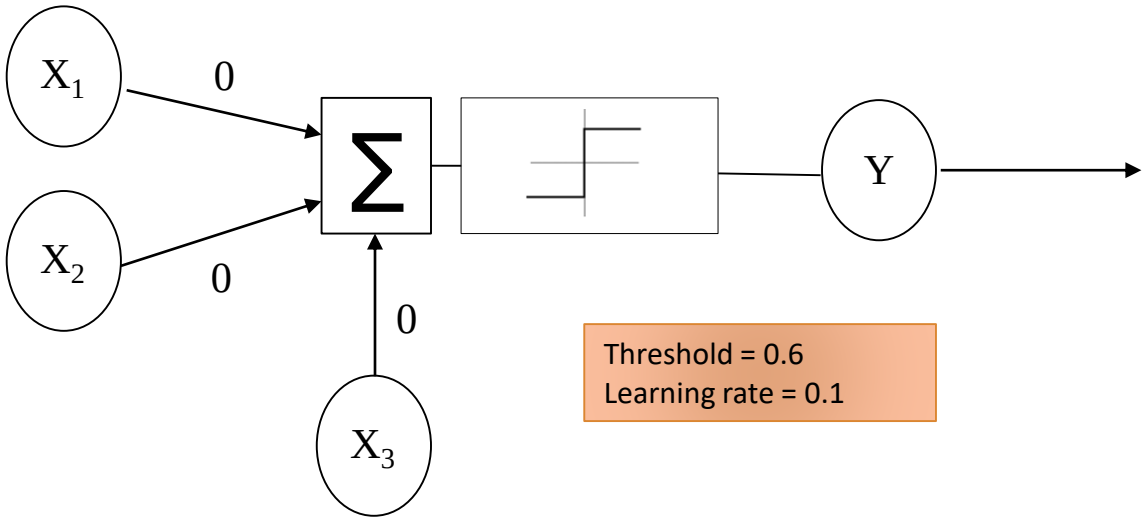
X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.2	-0.1	0.7	0.7	1	0	$\mp$0.0
0	1	1	-0.2	-0.1	0.7	0.6	1	0	$\mp$0.0
1	0	1	-0.2	-0.1	0.7	0.5	0	1	+0.1
1	1	1	-0.1	-0.1	0.8	0.6	1	-1	-0.1



TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

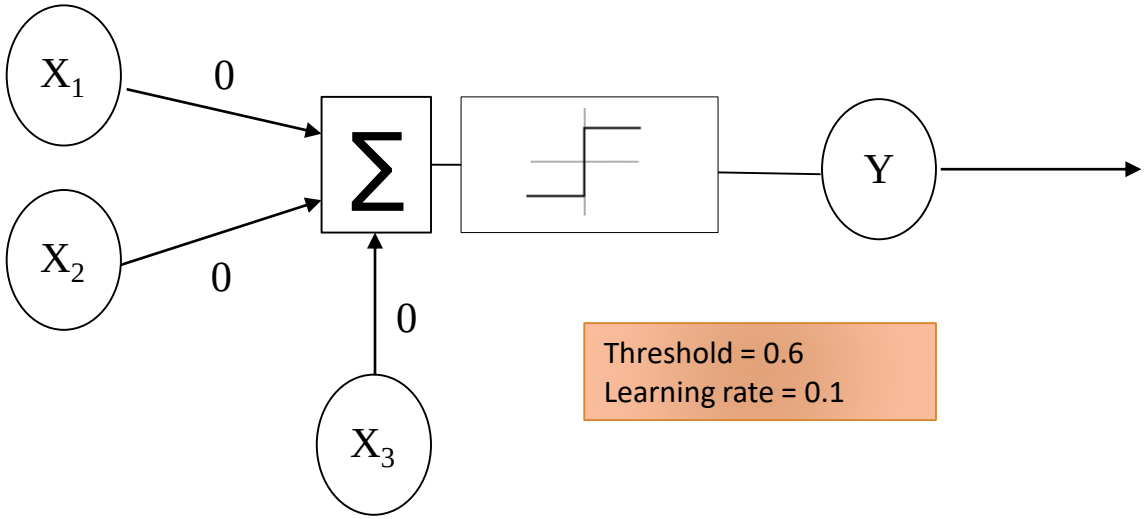
X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.2	-0.2	0.7	0.7	1	0	$\mp$0.0
0	1	1	-0.2	-0.2	0.7	0.5	0	1	+0.1
1	0	1							
1	1	1							



TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.2	-0.2	0.7	0.7	1	0	$\mp$0.0
0	1	1	-0.2	-0.2	0.7	0.5	0	1	+0.1
1	0	1	-0.2	-0.1	0.8	0.6	1	0	$\mp$0.0
1	1	1	-0.2	-0.1	0.8	0.5	0	0	$\mp$0.0

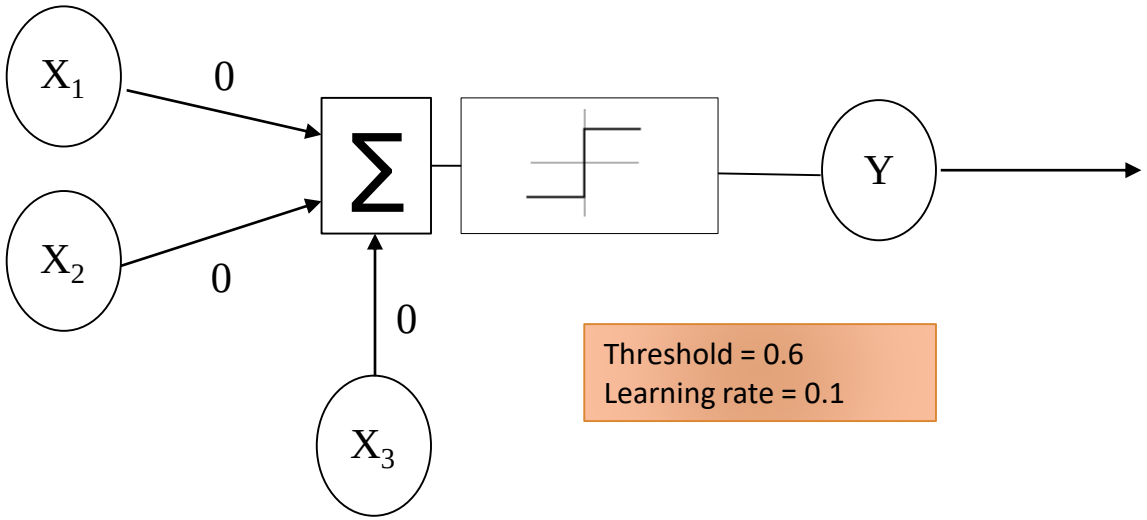


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.2	-0.1	0.8	0.8	1	0	∓0.0
0	1	1							
1	0	1							
1	1	1							

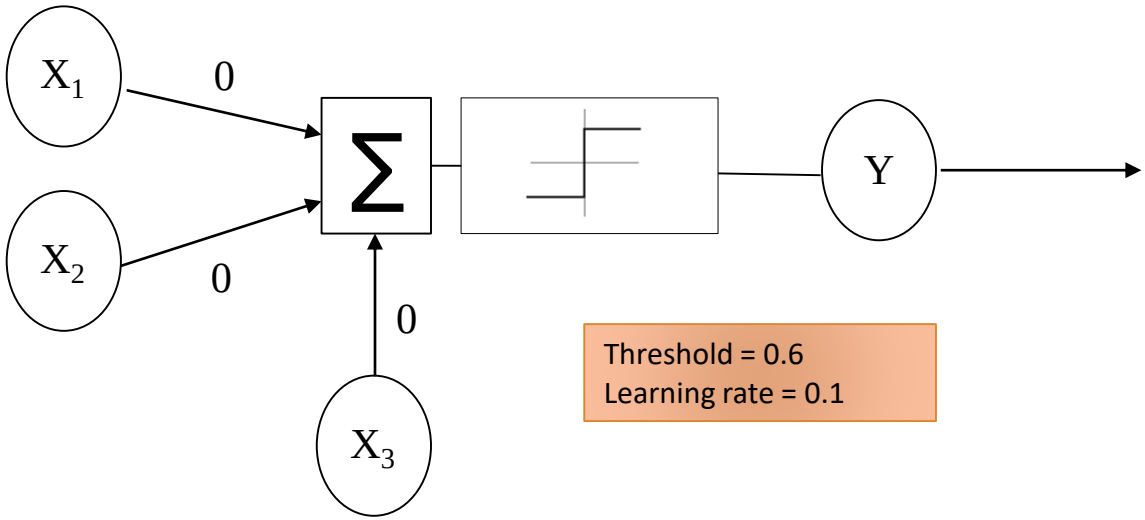


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.2	-0.1	0.8	0.8	1	0	∓0.0
0	1	1	-0.2	-0.1	0.8	0.7	1	0	∓0.0
1	0	1							
1	1	1							

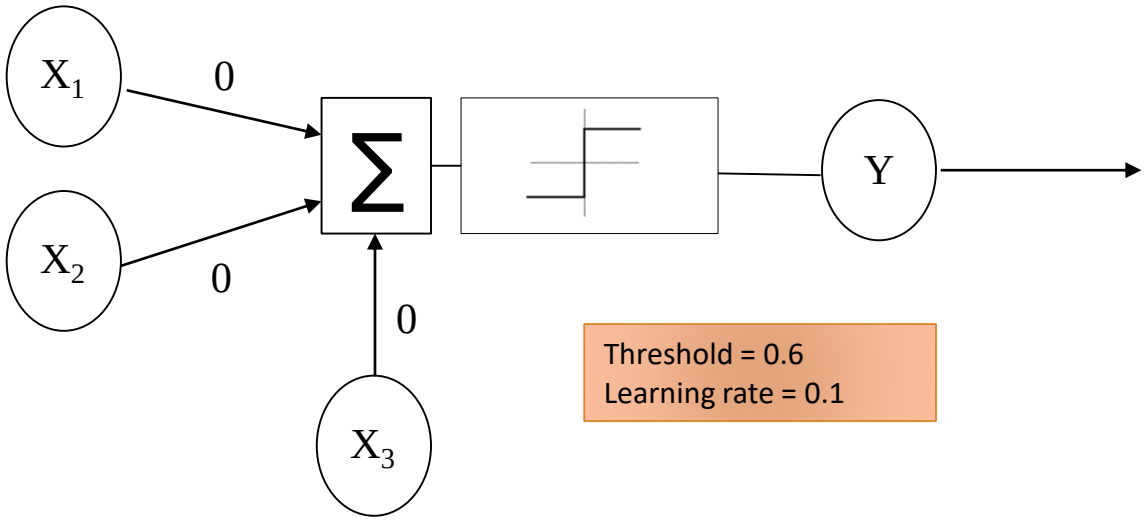


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.2	-0.1	0.8	0.7	1	0	$\mp$0.0
0	1	1	-0.2	-0.1	0.8	0.7	1	0	$\mp$0.0
1	0	1	-0.2	-0.1	0.8	0.6	1	0	$\mp$0.0
1	1	1							

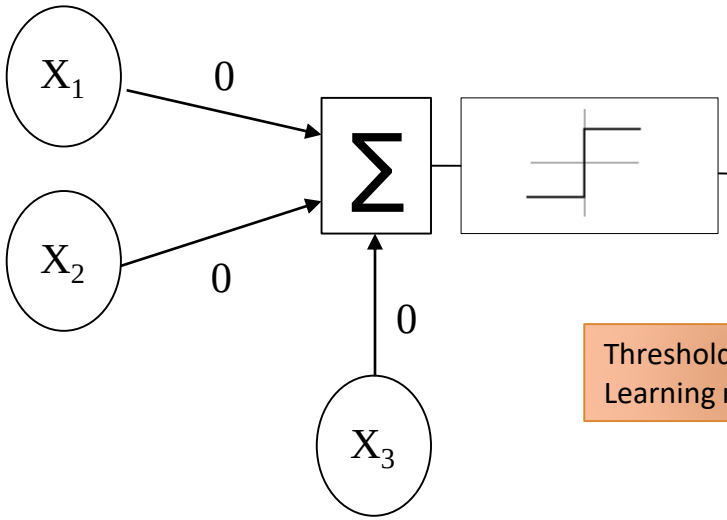


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.2	-0.1	0.8	0.7	1	0	$\mp$0.0
0	1	1	-0.2	-0.1	0.8	0.7	1	0	$\mp$0.0
1	0	1	-0.2	-0.1	0.8	0.6	1	0	$\mp$0.0
1	1	1	-0.2	-0.1	0.8	0.5	0	0	$\mp$0.0



Threshold = 0.6
Learning rate = 0.1