Neural Networks Fundamentals of AI (AE1FAI)

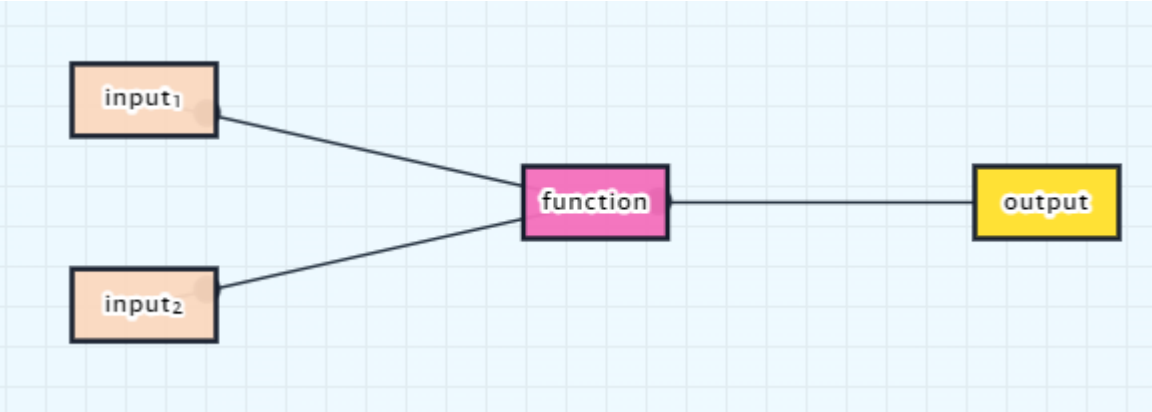
Slides created by Dr Rong Qu & Dr John Drake
Modified by Dr Huan Jin

OUTLINE

- Computational graph representation
- Architecture of Neural Networks
- Deep learning
- Perceptron
- Train a neural network

A COMPUTATIONAL GRAPH

- An input node where data is fed into the graph,
- A function node where the input data is processed,
- An output node where the result of the computation is produced.



LINEAR REGRESSION

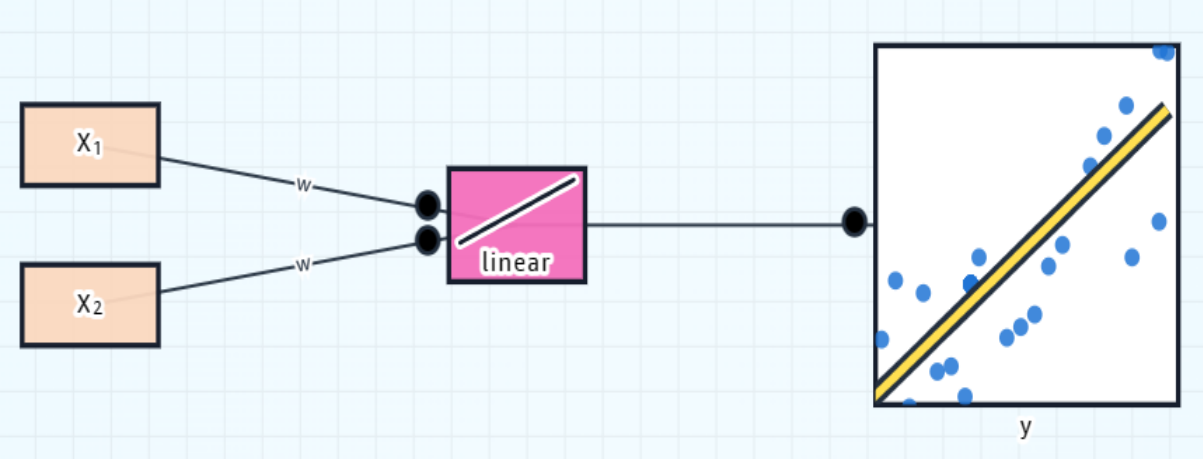
Represent Linear Regression as a computational graph:

1. Add weights to each edge connecting the input nodes to the function nodes.

$$y = w_0 + w_1 X_1$$

2. Update our function to represent a linear function of a weighted sum:

$$\text{linear} = \sum_{i=1}^{n} w_i X_i$$

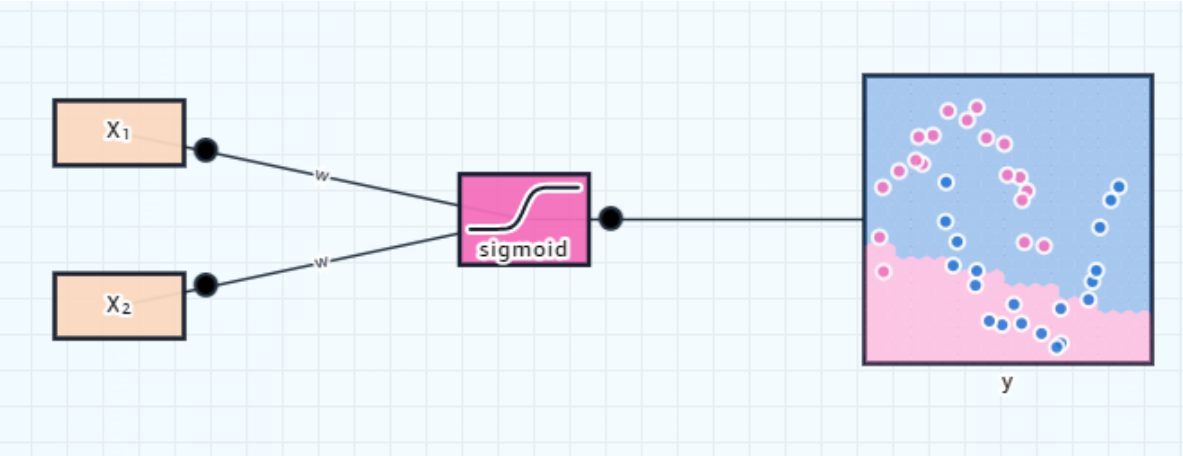


LOGISTIC REGRESSION

Represent Logistic Regression as a computational graph:

1. changing the linear function to a sigmoid function:

$$\text{sigmoid} = \frac{1}{1 + e^{-w_i X_i}}$$

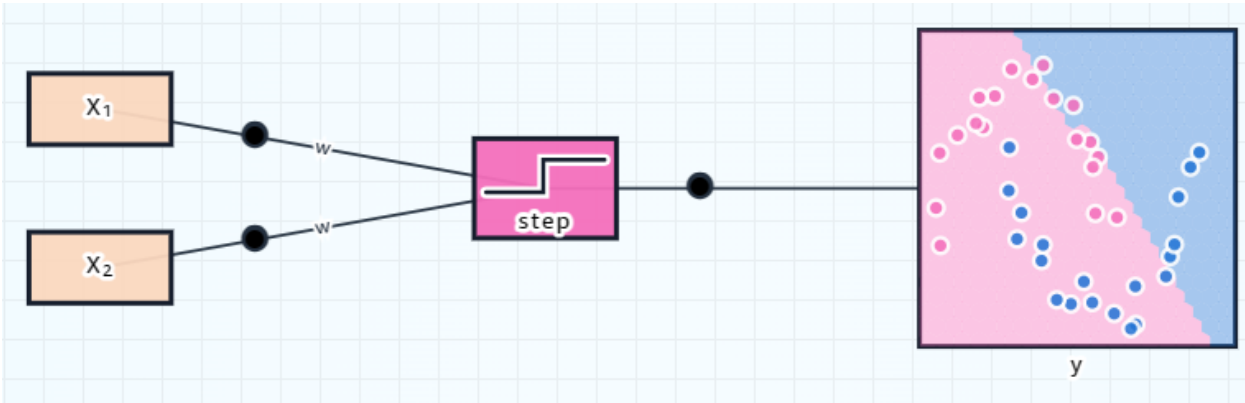


PERCEPTRON

Represent Perceptron as a computational graph:

1. Switch logistic function to a step function:

$$\text{step} = \begin{cases} +1, \text{if } w_i X_i \geq 0 \\ -1, \text{if } w_i X_i < 0 \end{cases}$$

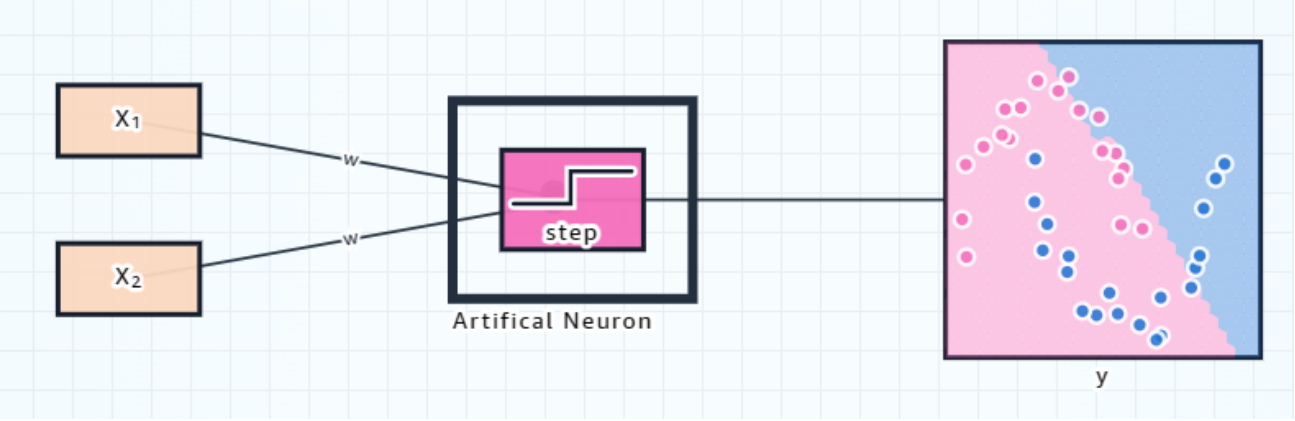


ACTIVATION FUNCTIONS & ARTIFICIAL NEURONS

In a neural network, this function node we're changing is very special - we call it an artificial neuron.

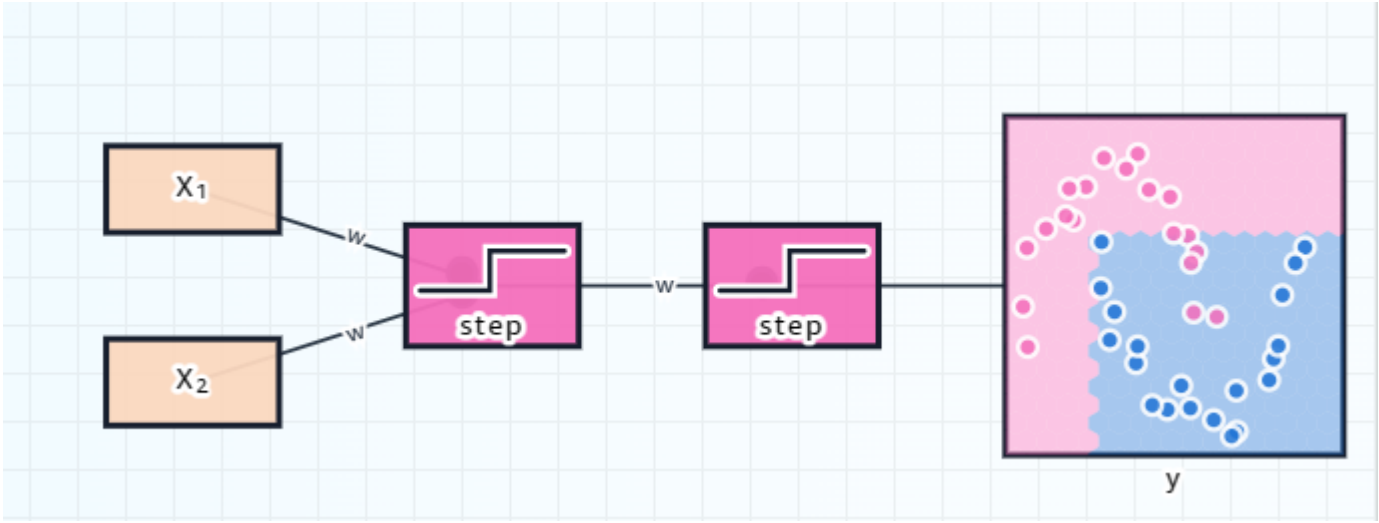
An artificial neuron is a fundamental computational element that receives inputs, performs a weighted operation on these inputs, and passes the result through a function.

In neural networks, these functions *must* be non-linear, and are referred to as activation functions.



NEURAL NETWORKS

The original neural networks were called multilayer perceptrons because they were composed of layers of perceptrons (artificial neurons with step functions) feeding one into another!

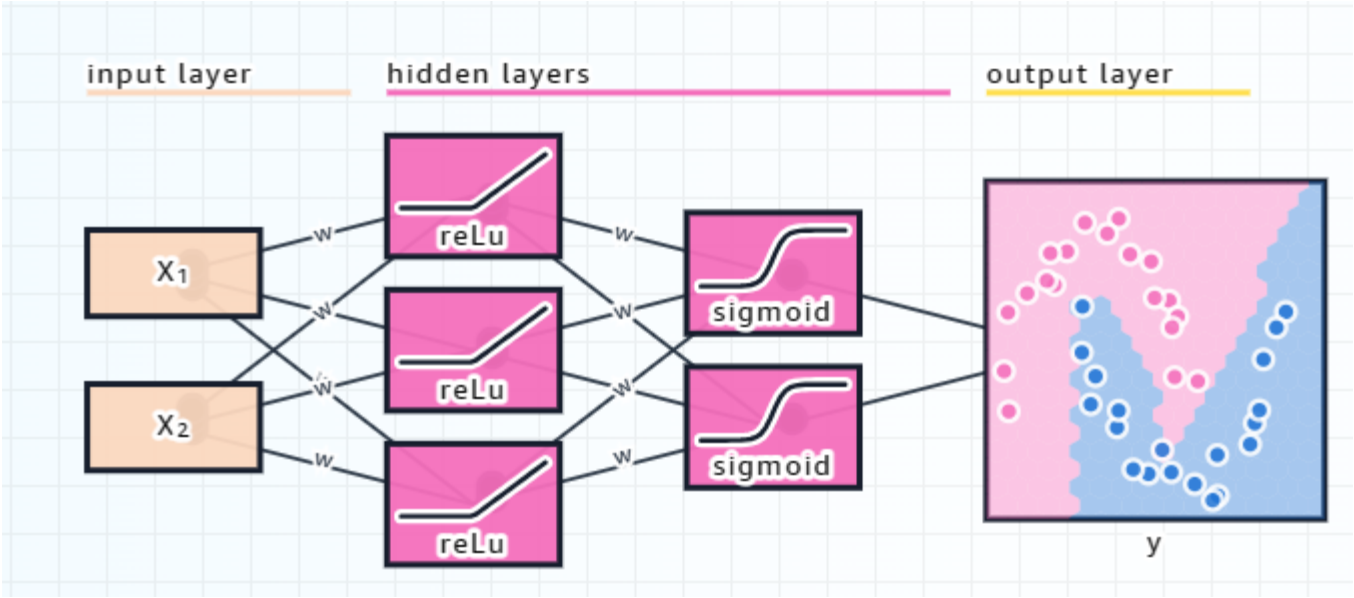


ARCHITECTURE

A neural network architecture consists of three layer types:

- input layer: A layer with a node for each network input.
- hidden layer(s): A layer full of artificial neurons.
- output layer: A layer representing the network's output.

There should only be one input and one output layer, but there may be an arbitrary number of hidden layers.



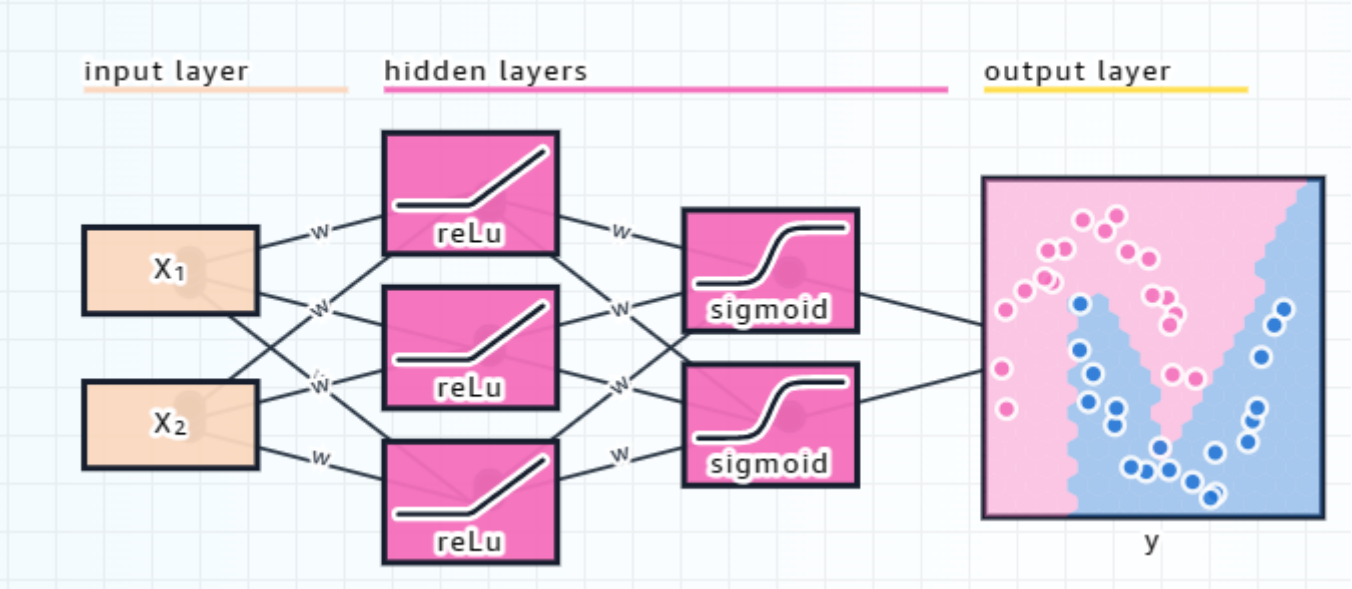
ARCHITECTURE

Hidden layer

- Output of hidden units serves as input for units in next layer
- Too many hidden layers or too many units per layer can lead to overfitting.

Output layer

- Number of output units depend on task
- Different activation functions for output layer and hidden layers possible.



ACTIVATION FUNCTIONS

Activation functions are at the heart of artificial neurons in a neural network.

These crucial components introduce non-linearity into the model, transforming the weighted inputs to generate an output.

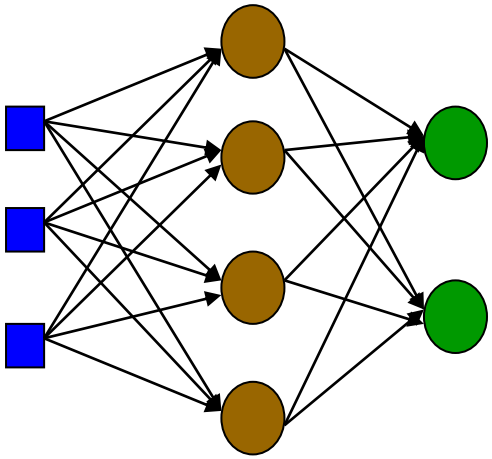
The non-linear nature of these functions is essential for neural networks to learn from complex data.

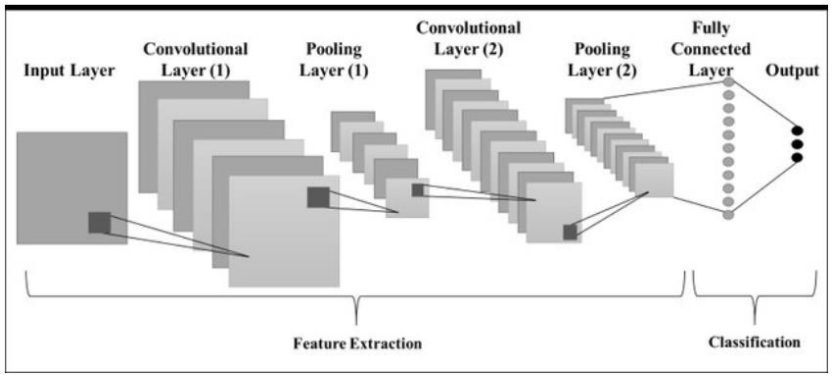
Name	Plot	Function	Description
Sigmoid (logistic)		$f(x) = \frac{1}{1 + e^{-x}}$	Squashes input to (0, 1).
Hyperbolic Tangent (tanh)		$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	Squashes input to (-1, 1).
Rectified Linear Unit (ReLu)		$f(x) = \begin{cases} 0, \text{if } x < 0 \\ x, \text{if } x \geq 0 \end{cases}$	Only keep positive values.
Step Function (Perceptron) [i]		$f(x) = \begin{cases} +1, \text{if } x \geq 0 \\ -1, \text{if } x < 0 \end{cases}$	Returns only -1 or 1 (neuron fires or doesn't fire).

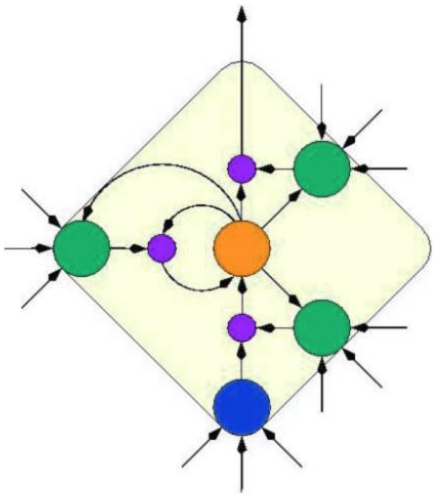
NEURAL NETWORK ARCHITECTURES

Three different classes of network architectures

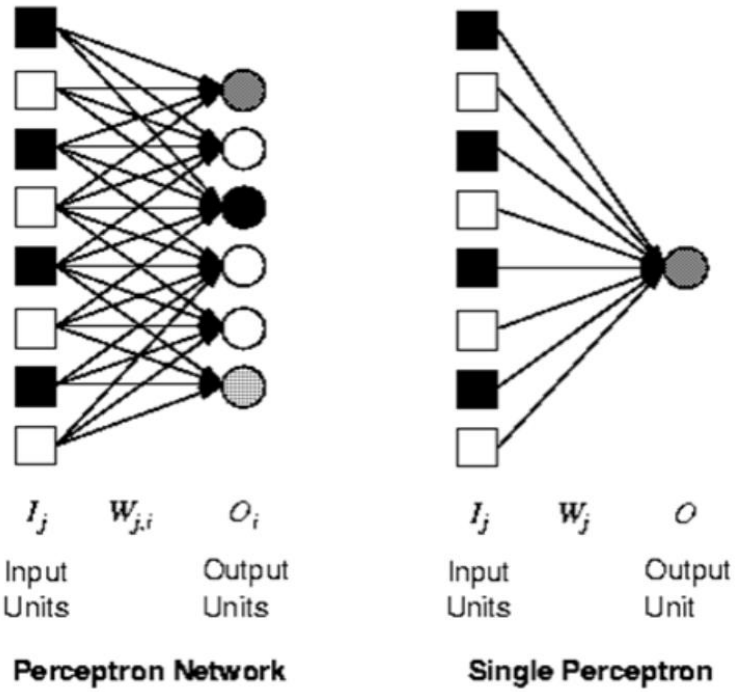
- Artificial neural networks(ANN)
- Convolutional neural network (CNN): Images
- Recurrent neural network (RNN): Text, language







PERCEPTRON



Synonym for Single-Layer, Feed-Forward Network

- First Studied in the 1950's

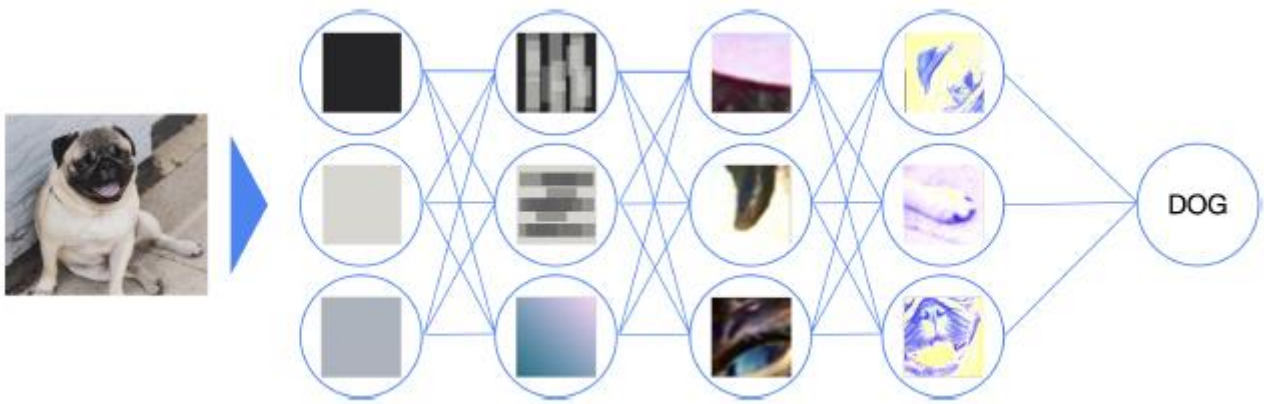
DESIGN OF NN

Input and output layer:

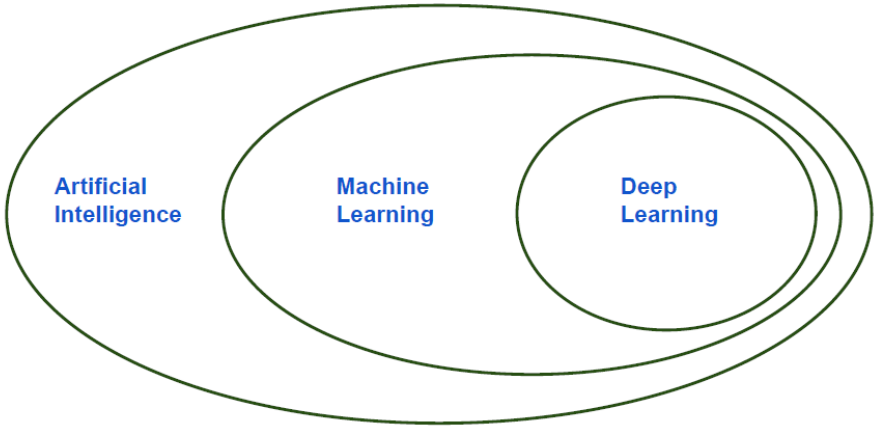
- The input and output layer will be selected for the specific problem, but the hidden layer is fairly arbitrary.

Effect of hidden layers:

- **Wide**: having many neurons in a given hidden layer;
- **Deep**: having many hidden layers in the network;
- Each layer adds degree of non-linearity
- Balancing neuron count optimizes performance; more neurons enable complex learning, at the cost of the risk of overfitting and more computational cost.



DEEP LEARNING



Deep learning itself is not new:

- Neural networks have been around since the 70s.
- Deep neural networks, i.e., networks with multiple hidden layers, are not much younger.

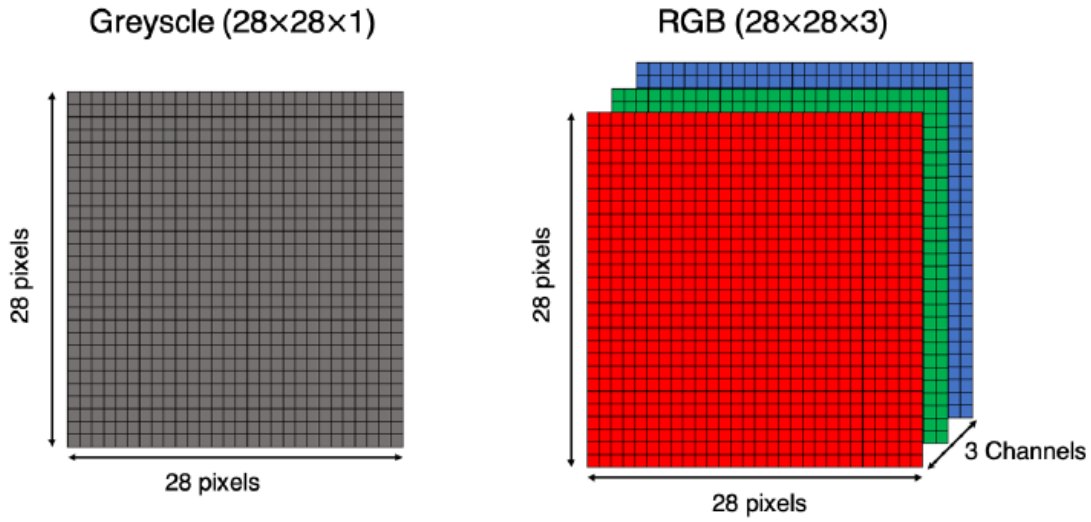
Why everybody is talking about deep learning now:

- Specialized, powerful hardware allows training of huge neural networks to push the state-of-the-art on difficult problems.
- Large amount of data is available.
- Special network architectures for image/text data.
- Better optimization and regularization strategies.

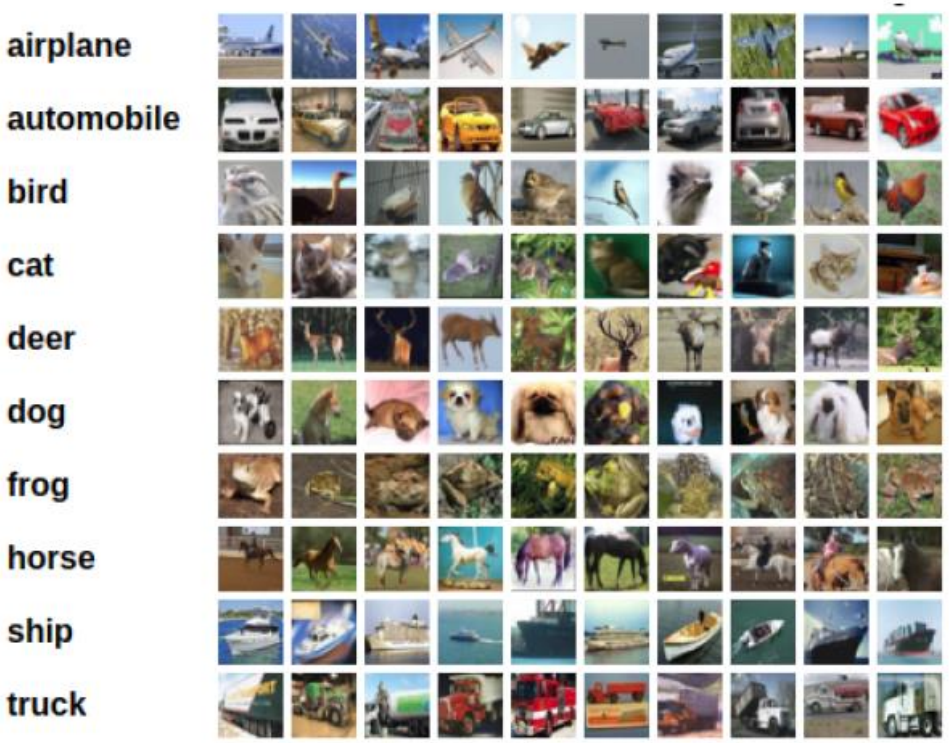
POSSIBLE USE-CASE: IMAGES

- **High Dimensional**: A color image with 255×255 (3 Colors) pixels already has 195075 features.
- **Informative**: A single pixel is not meaningful in itself.
- **Training Data**: Depending on applications huge amounts of data are available.

Architecture: **C**onvolutional **N**eural **N**etworks (CNN)



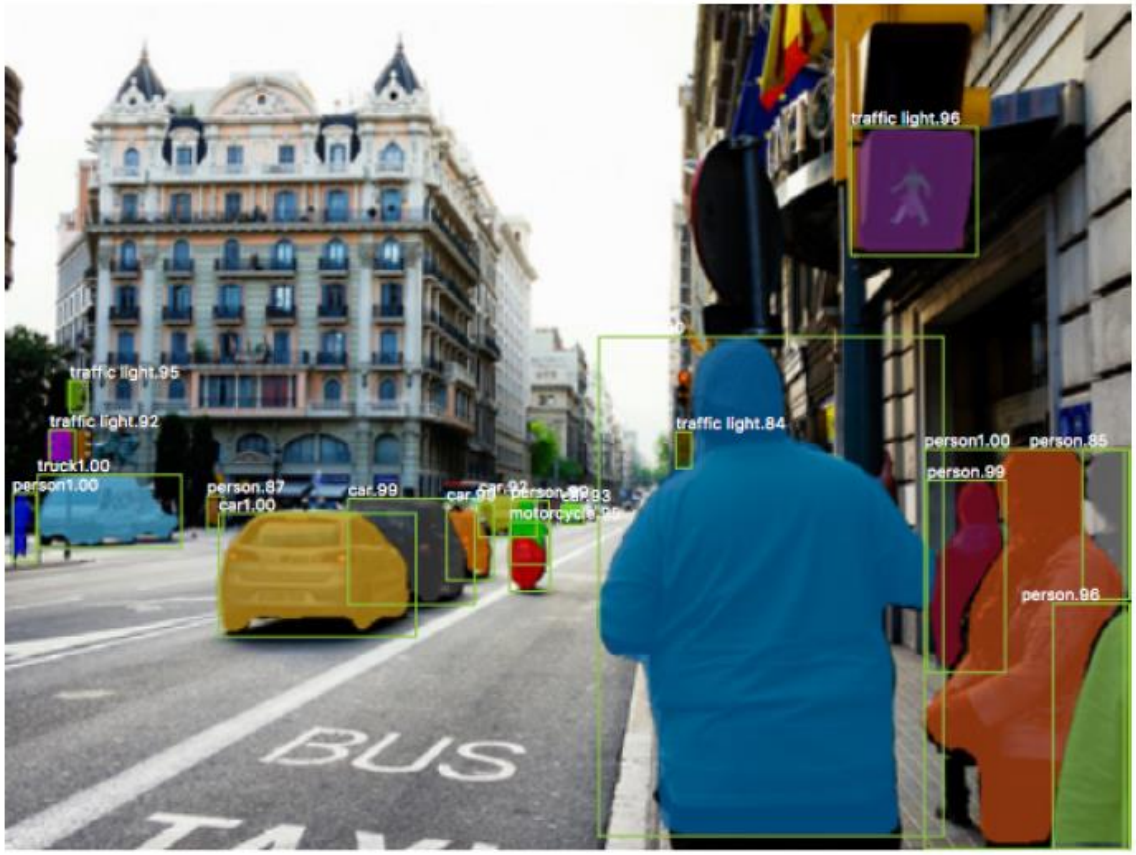
POSSIBLE USE-CASE: IMAGES



Credit: Alex Krizhevsky (2009)

Image classification tries to predict a single label for each image.
CIFAR-10 is a well-known dataset used for image classification. It consists of 60, 000 32*x*32 color images containing one of 10 object classes, with 6000 images per class.

POSSIBLE USE-CASE: IMAGES



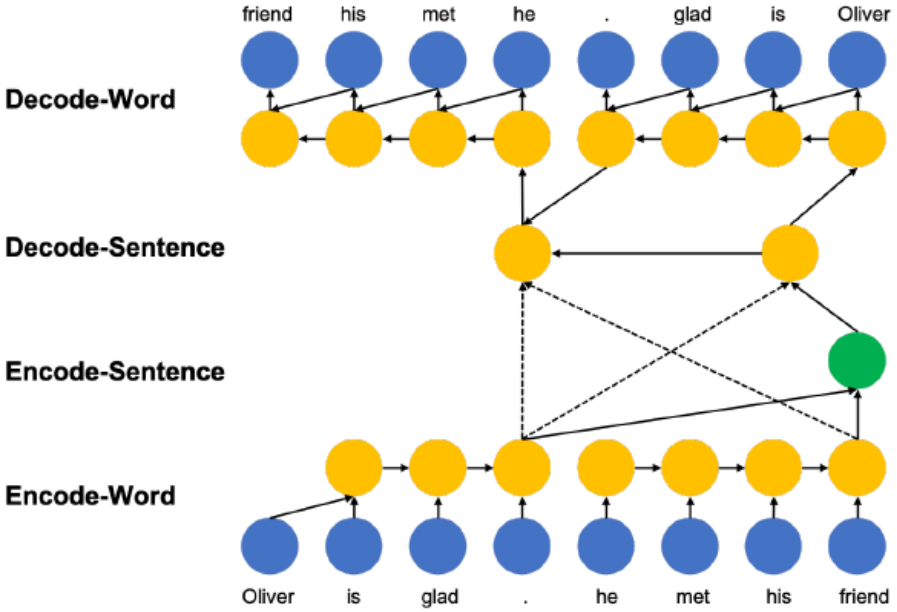
Credit: Kaiming He (2017)

Object Detection Mask R-CNN is a general framework for instance segmentation, that efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance.

POSSIBLE USE-CASE: TEXT

- **High Dimensional**: Each word can be a single feature (300000 words in the German language).
- **Informative**: A single word does not provide much context.
- **Training Data**: Huge amounts of text data available.

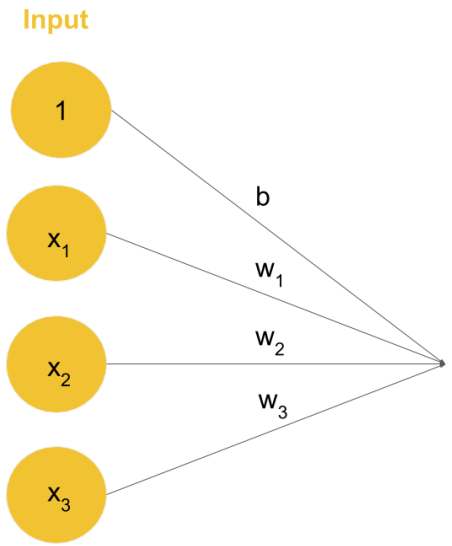
Architecture: **R**ecurrent **N**eural **N**etworks (RNN)



POSSIBLE USE-CASE: TEXT

Machine Translation (e.g. google translate) Neural machine translation exploits neural networks to predict the likelihood of a sequence of words, typically modeling entire sentences in a single integrated model.

A SINGLE NEURON
BASIC ARCHITECTURE OF NN



- Weights **w** are connected to edges from the input layer.
- The bias term b
- $y\_in = x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + b$
  Weighted Sum

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

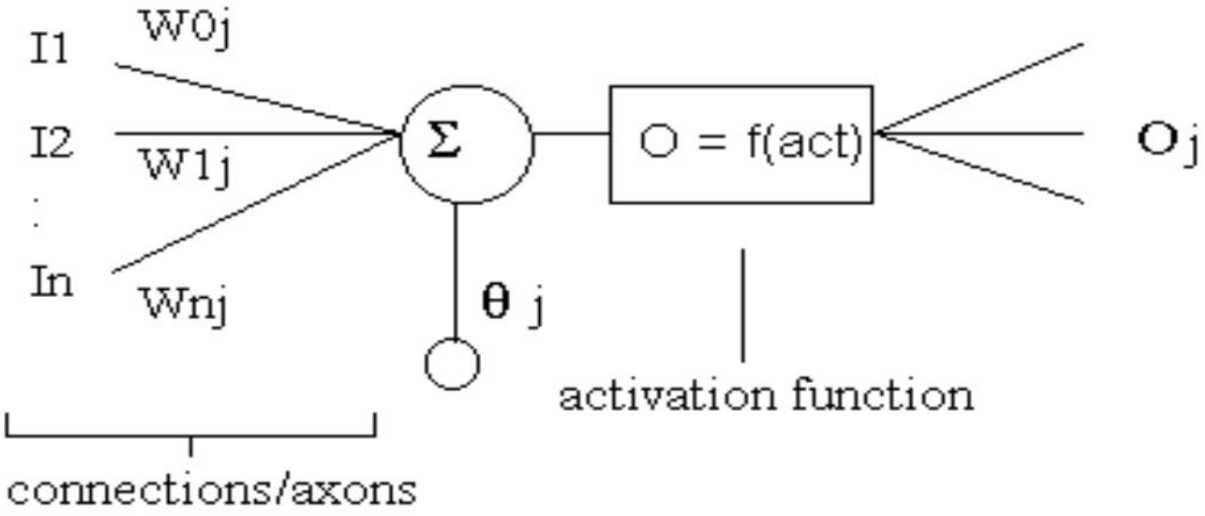
if *y_in* >= θ, *f(y_in)* = 1 (fire);
else *f(y_in)* = 0 (not fire)

Where *y_in* is the total input signal received (weighted sum);
θ is the **threshold** for *Y*

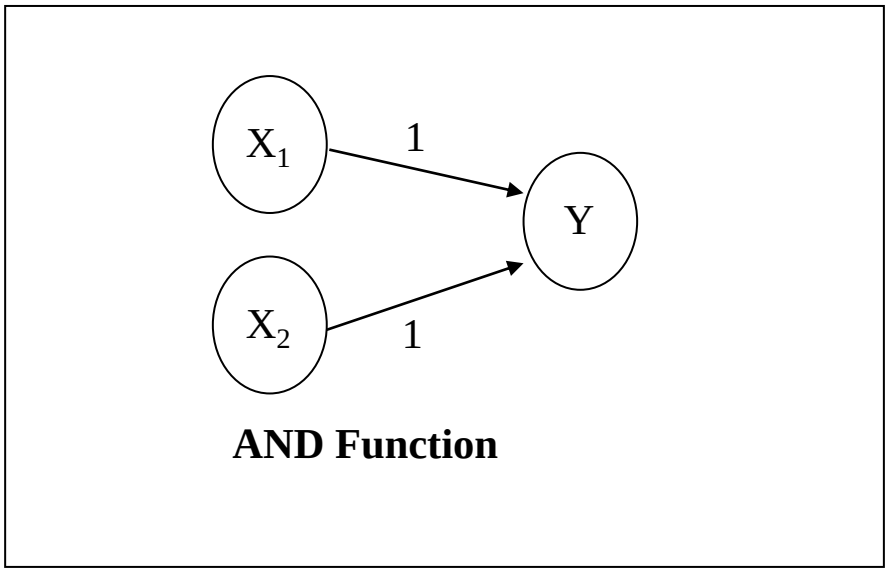
THE FIRST NEURAL NETWORKS

Using NN, we can model logic functions

Let's look at 4 logic functions



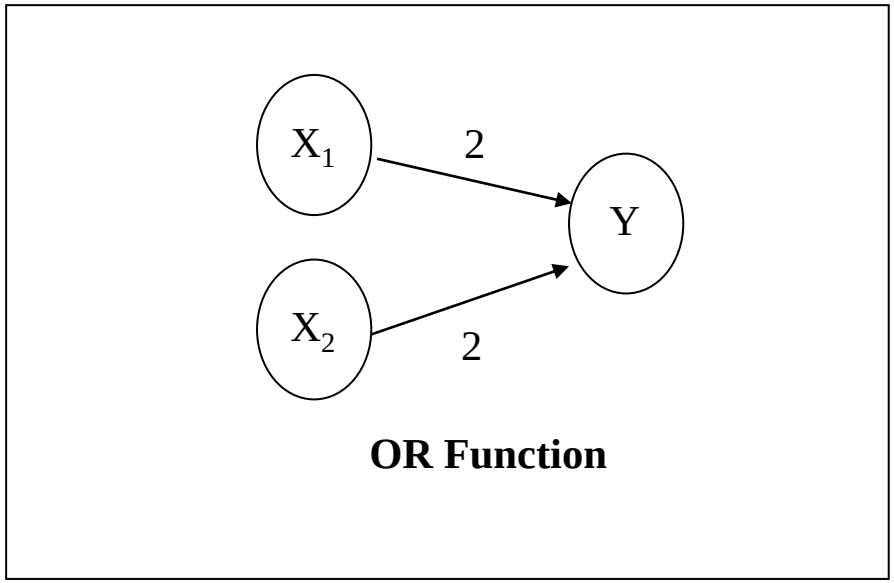
THE FIRST NEURAL NETWORKS



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

Threshold(Y) = 2

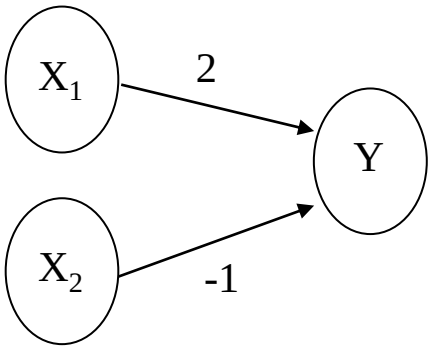
THE FIRST NEURAL NETWORKS



Threshold(*Y*) = 2

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

THE FIRST NEURAL NETWORKS



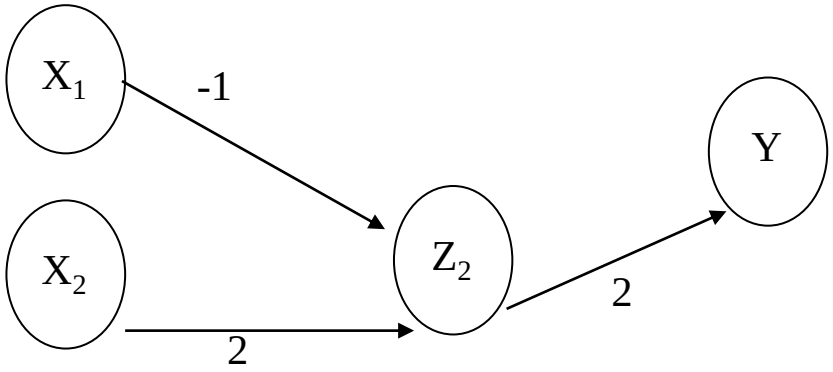
AND NOT Function

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)

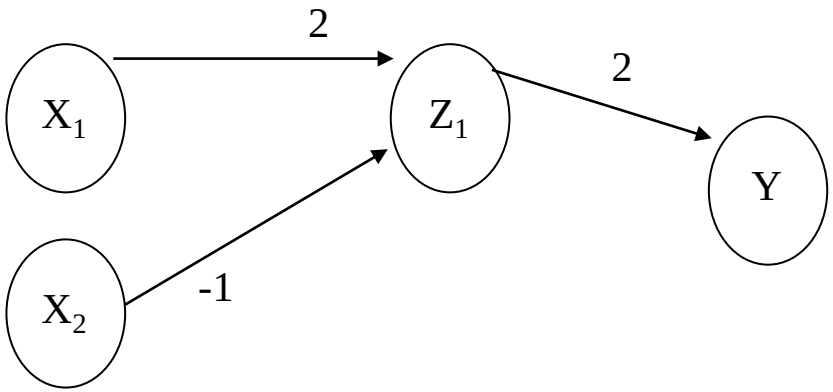


XOR		
X1	X2	Y
1	1	0
1	0	1
0	1	1
0	0	0

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)

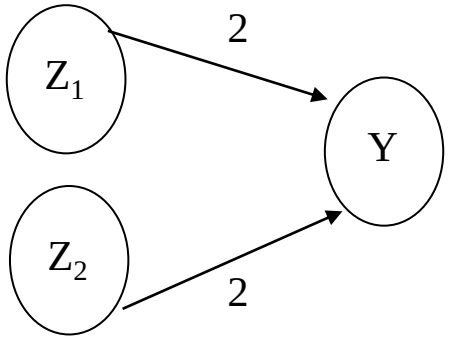


XOR		
X1	X2	Y
1	1	0
1	0	1
0	1	1
0	0	0

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)

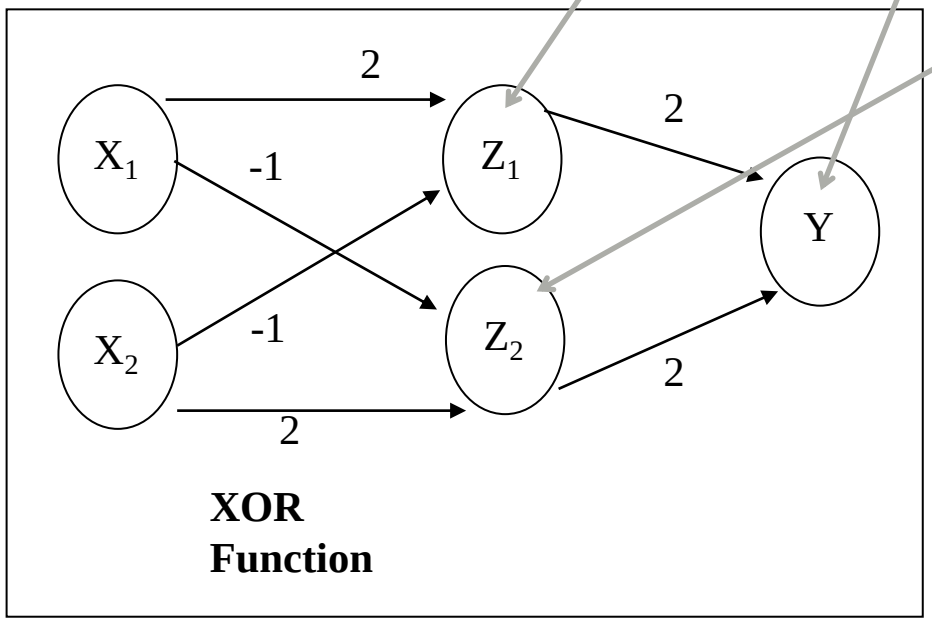


XOR		
X1	X2	Y
1	1	0
1	0	1
0	1	1
0	0	0

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

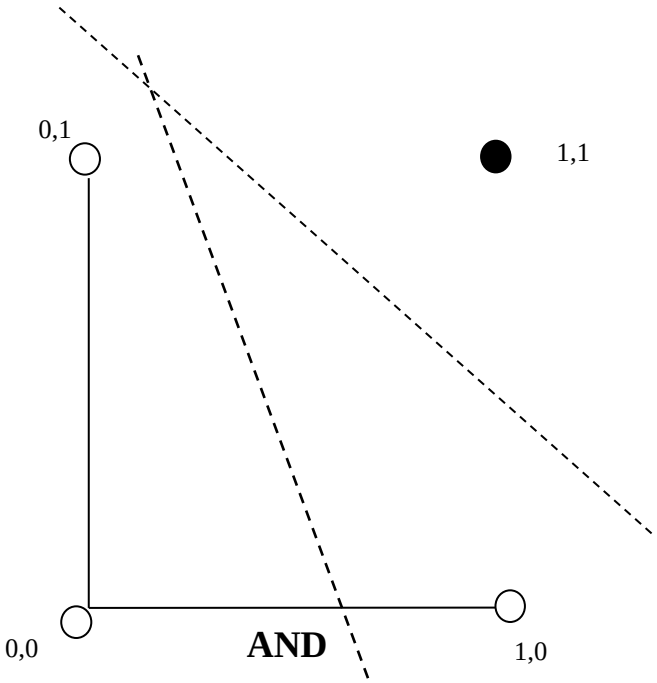
THE FIRST NEURAL NETWORKS

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

LINEAR SEPARABILITY

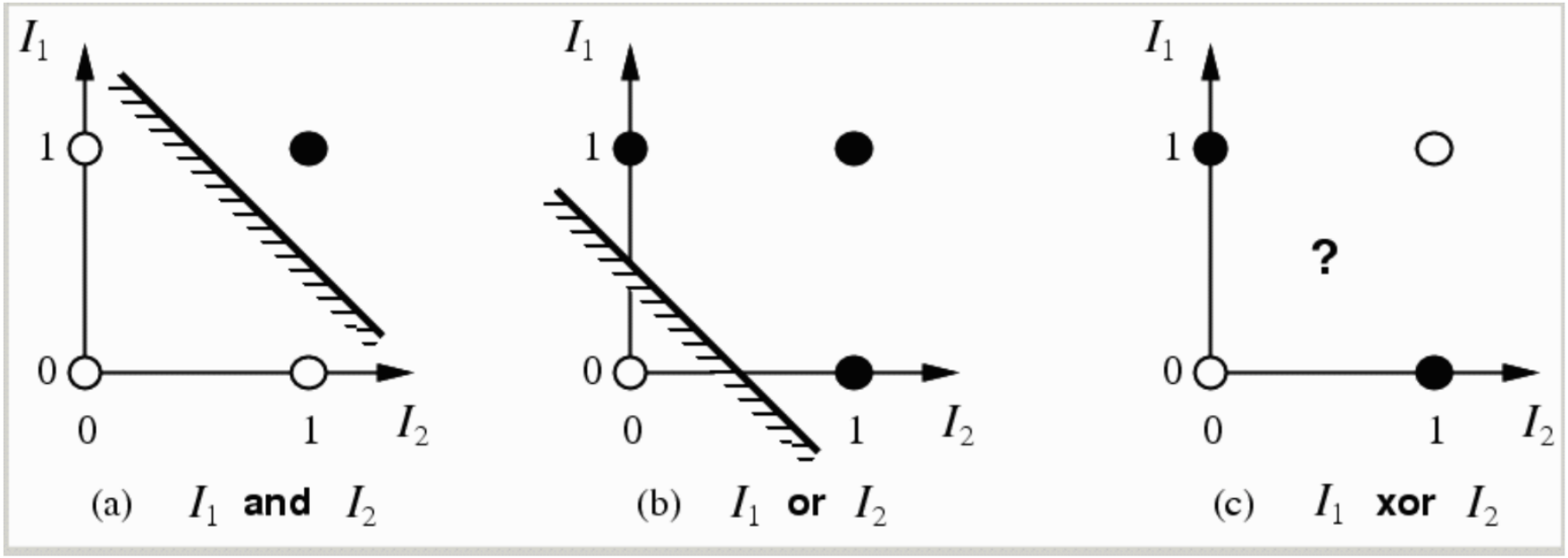


	AND			
Input 1	0	0	1	1
Input 2	0	1	0	1
Output	0	0	0	1

- 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

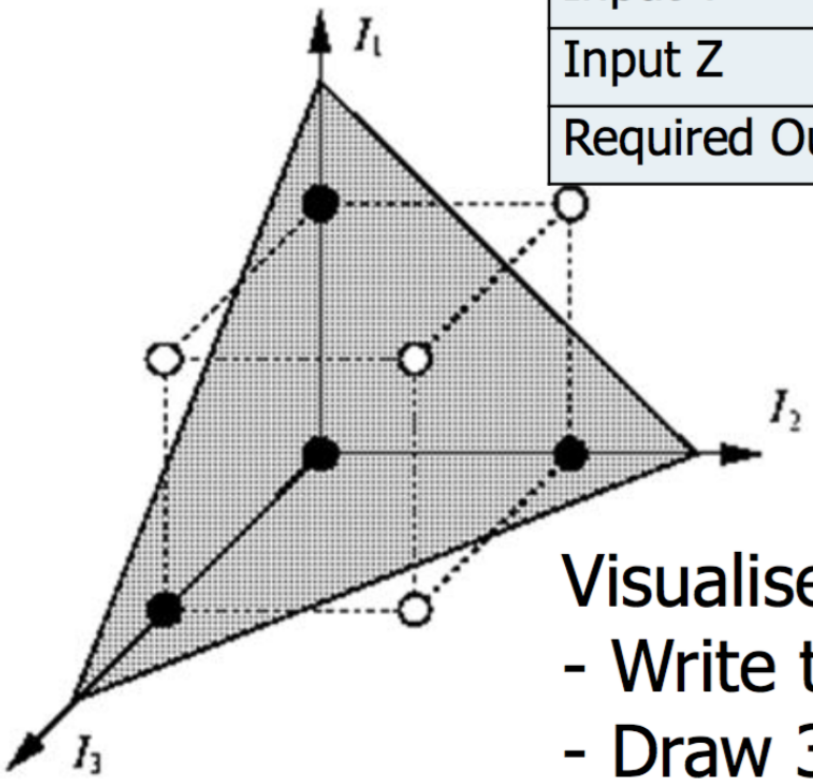
Minsky & Papert



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

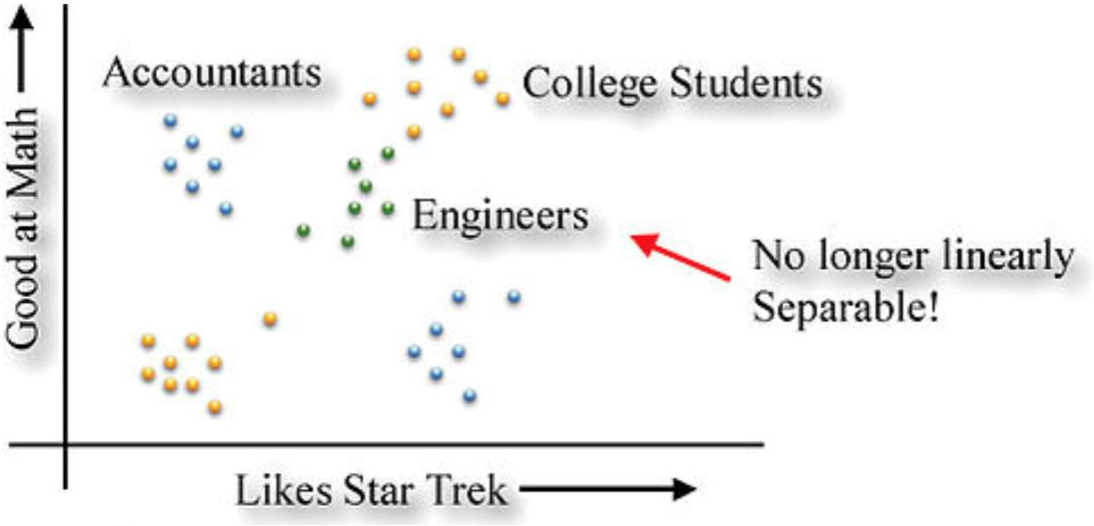
LINEAR SEPARABILITY

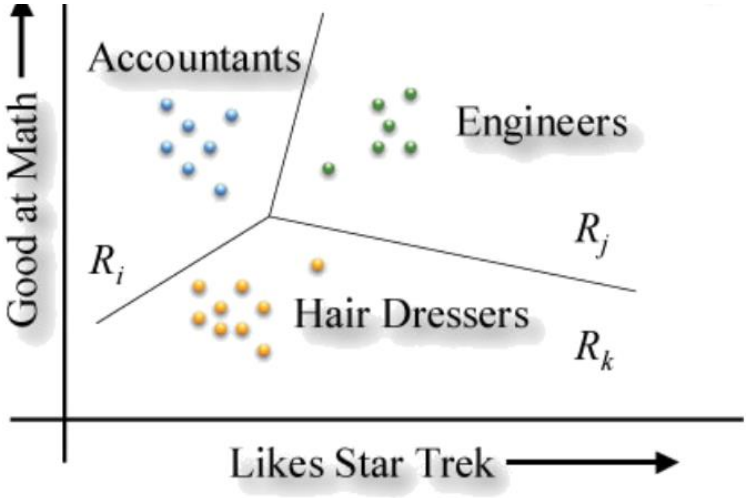
Input X	0	0	1	1	0	0	1	1
Input Y	0	1	0	1	0	1	0	1
Input Z	0	0	0	0	1	1	1	1
Required Output	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**0**



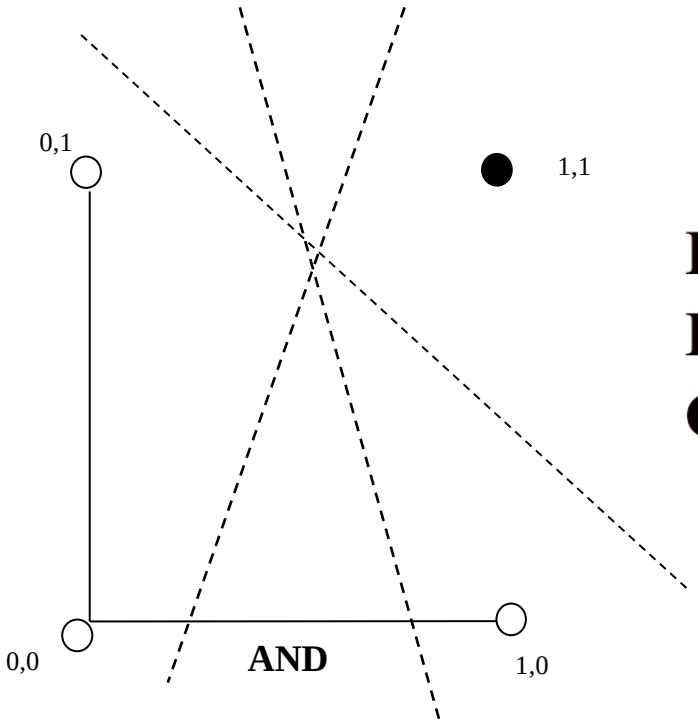
Visualise ANN with 3 inputs
- Write the truth table
- Draw 3D graphical representation

LINEAR SEPARABILITY



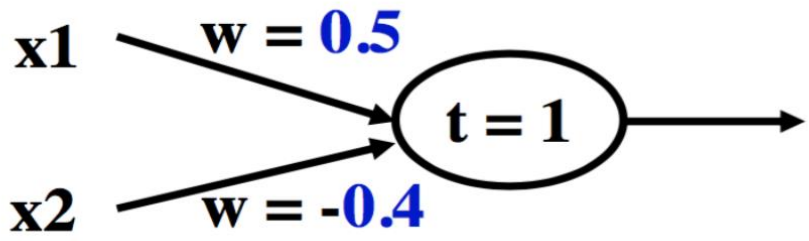


TRAINING A NN



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

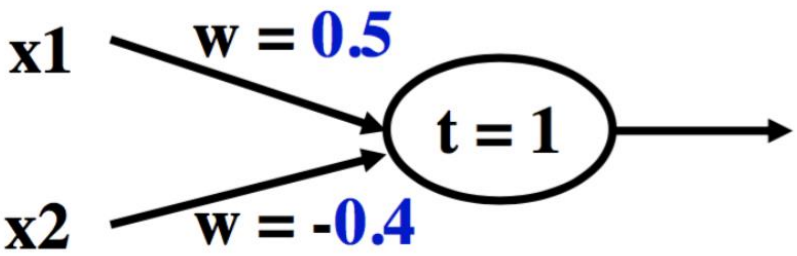
Training a NN



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

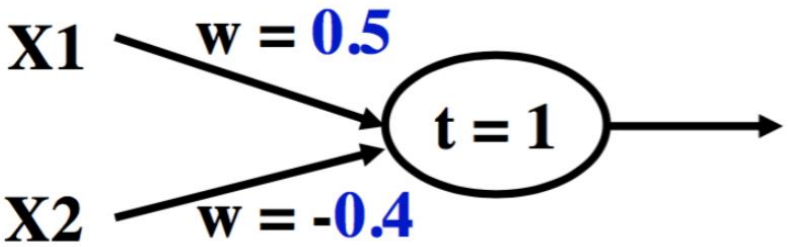
Training a NN

While epoch produces an error

Check next inputs (pattern) from epoch

End While

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



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

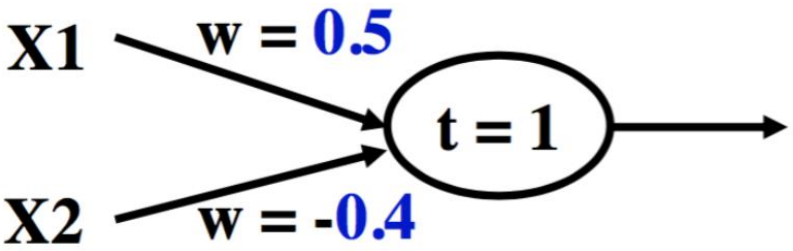
AND			
X1	X2	T	O
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	0

While epoch produces an error

Check next inputs (pattern) from epoch

Err = T – O

End While



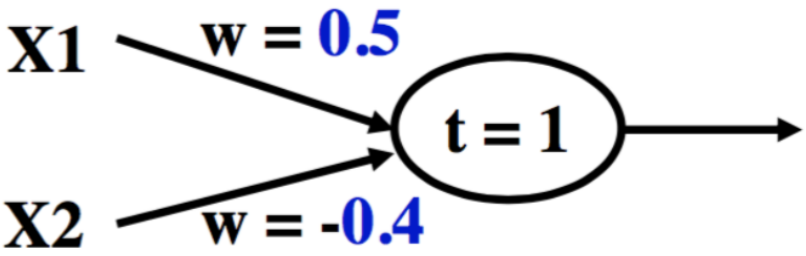
Training Value, T : The value that we require the network to produce. For example, with [0,0] for the AND function the training value will be 0.

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



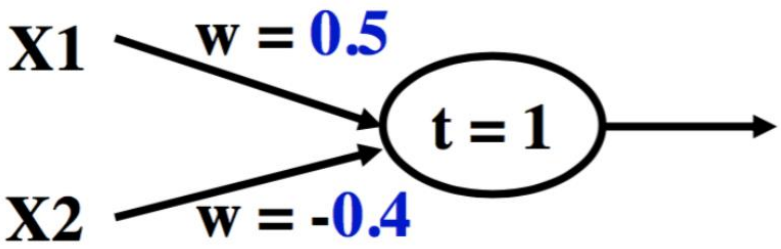
Error, Err : The amount the output by the network O differs from the training value T.

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

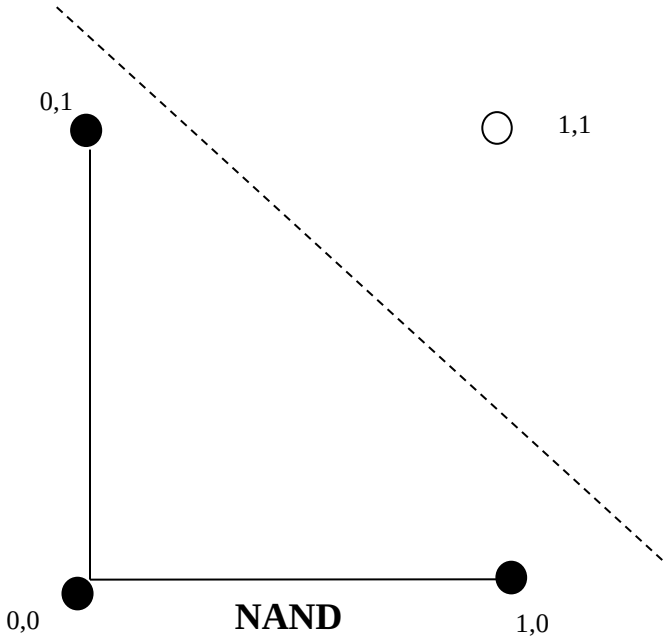
LR: The learning rate: how quickly the network converges. It is set by experimentation, typically 0.1

TRAINING A PERCEPTRON

Can we train a perceptron to learn logical NAND?

Training neural networks isn't easy! Even for small, relatively simple datasets such as those shown here, training a neural network can take some care and finesse.

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



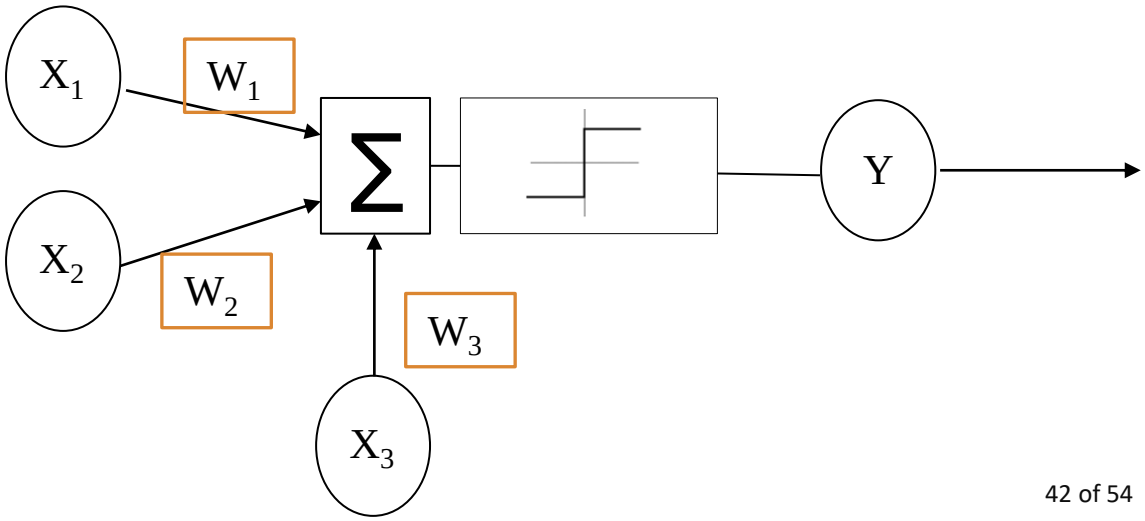
THE LEARNING PROCESS

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

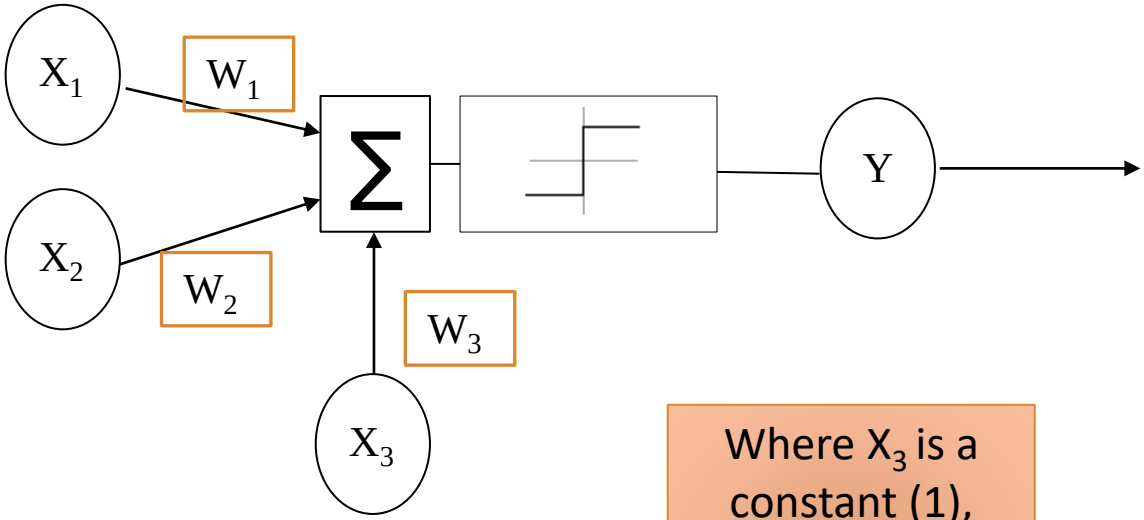


TRAINING A PERCEPTRON

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

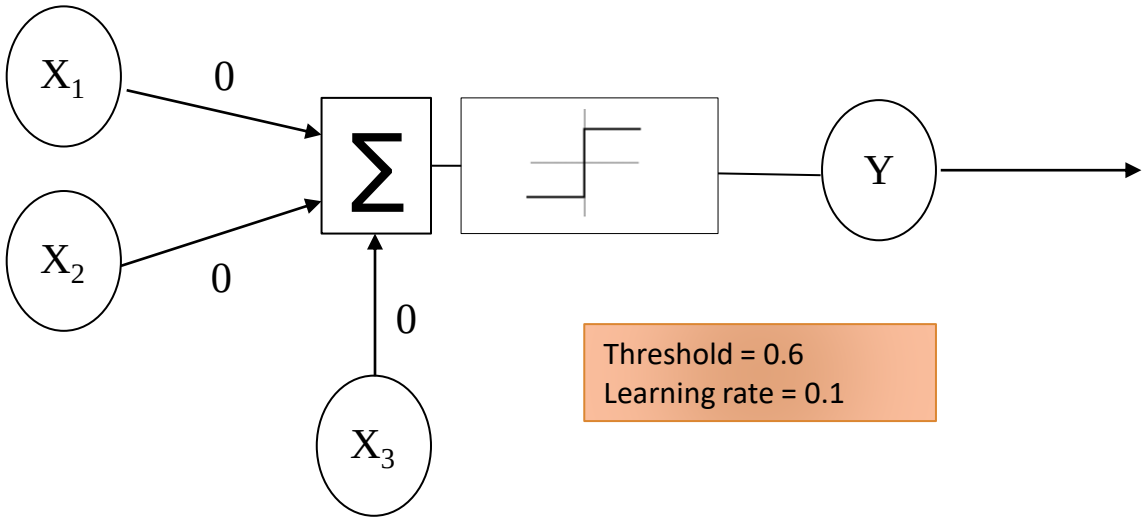


Where X_3 is a constant (1), weighting the perceptron within the network

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							

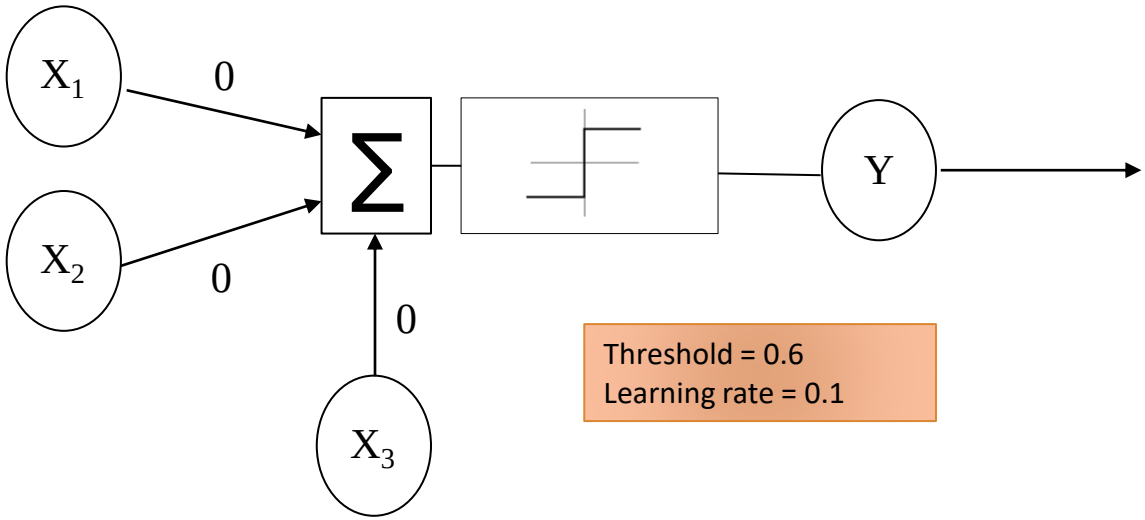


Threshold = 0.6
Learning rate = 0.1

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							

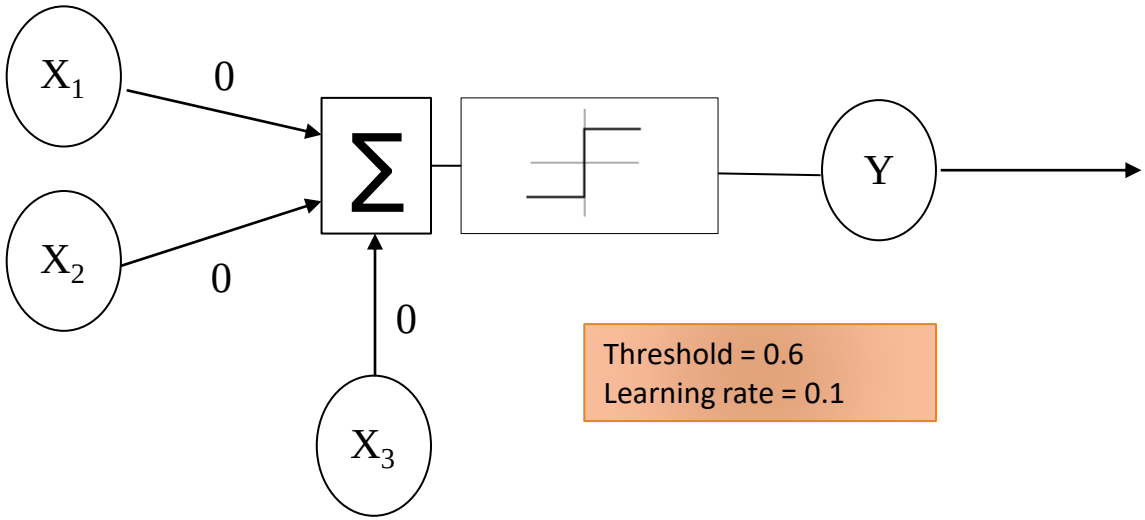


Threshold = 0.6
Learning rate = 0.1

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							

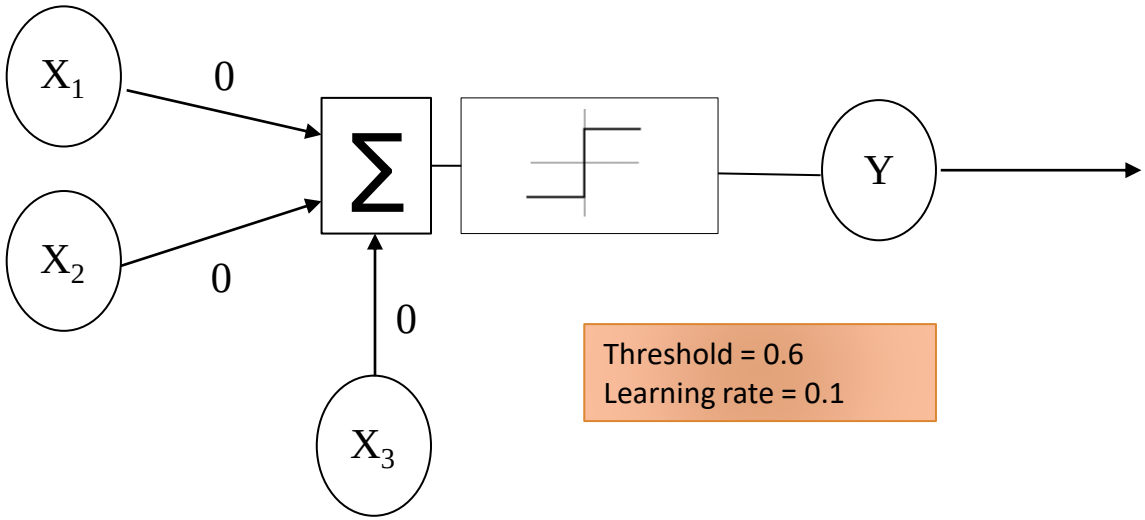


Threshold = 0.6
Learning rate = 0.1

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							

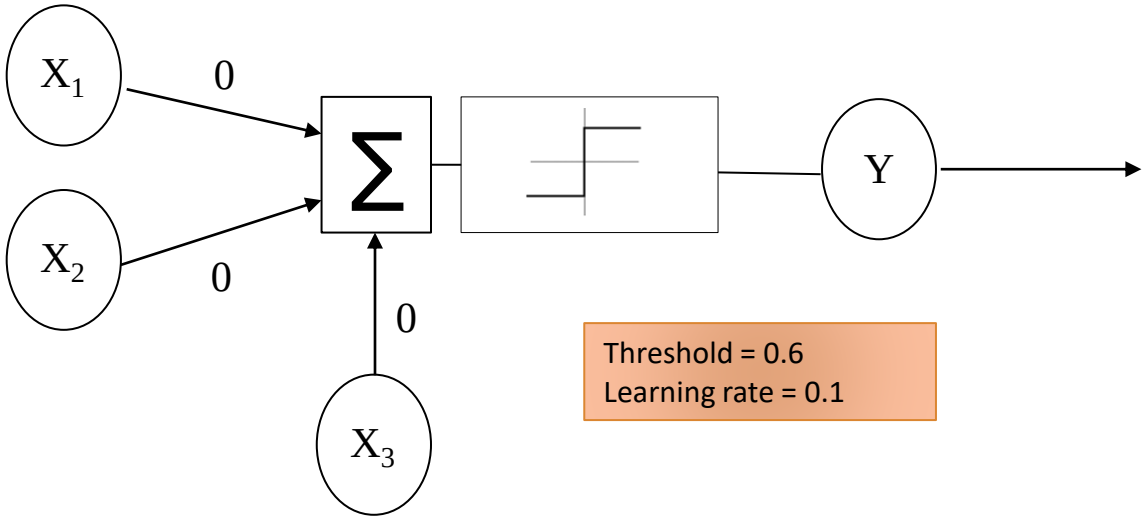


Threshold = 0.6
Learning rate = 0.1

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							

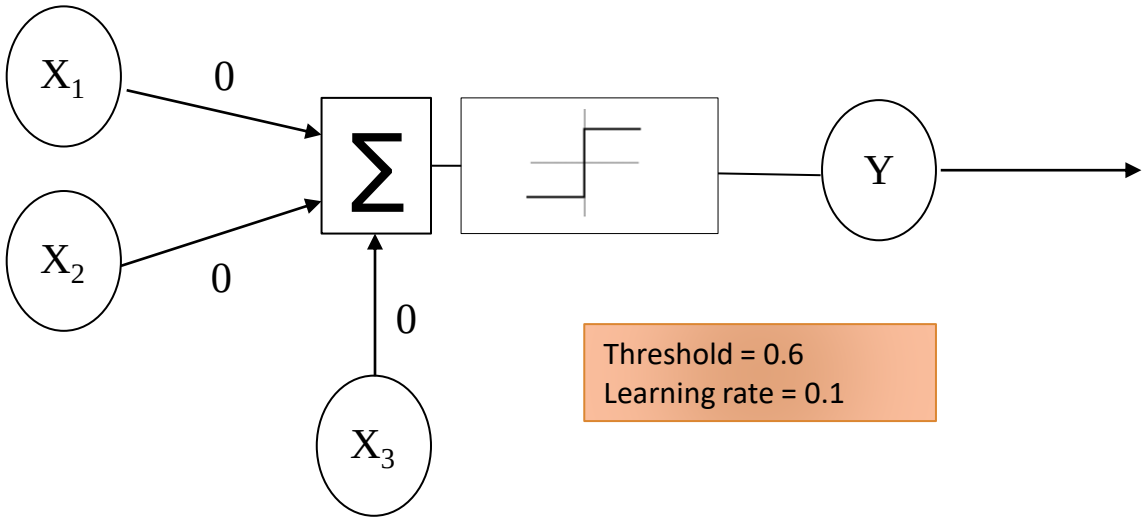


Threshold = 0.6
Learning rate = 0.1

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							

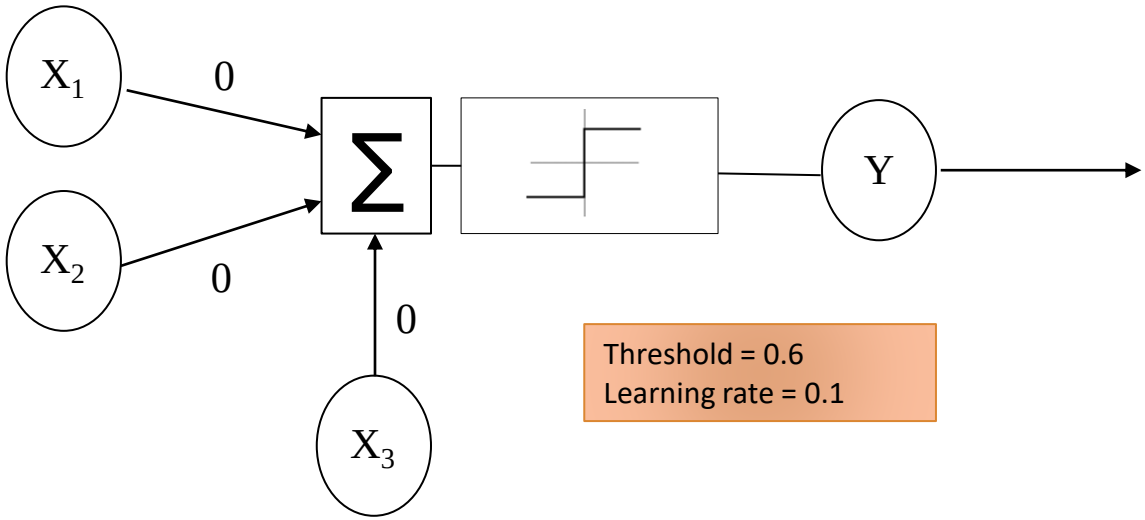


Threshold = 0.6
Learning rate = 0.1

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			

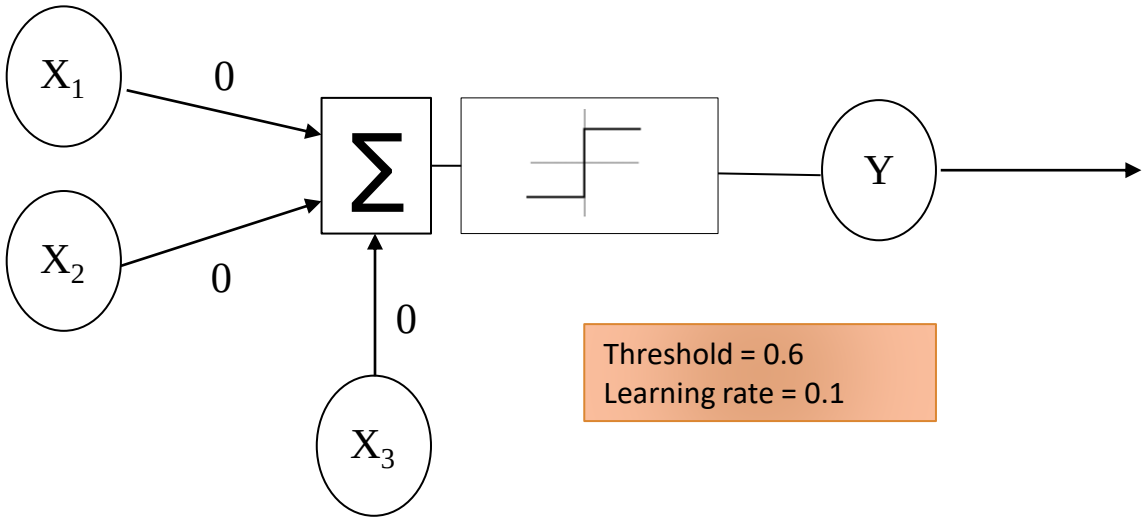


Threshold = 0.6
Learning rate = 0.1

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

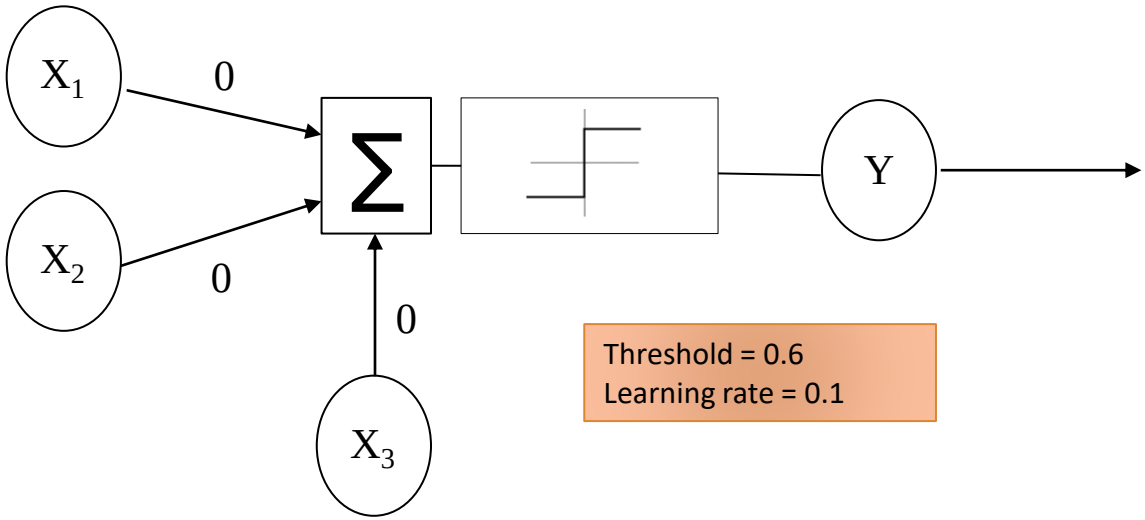


Threshold = 0.6
Learning rate = 0.1

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							

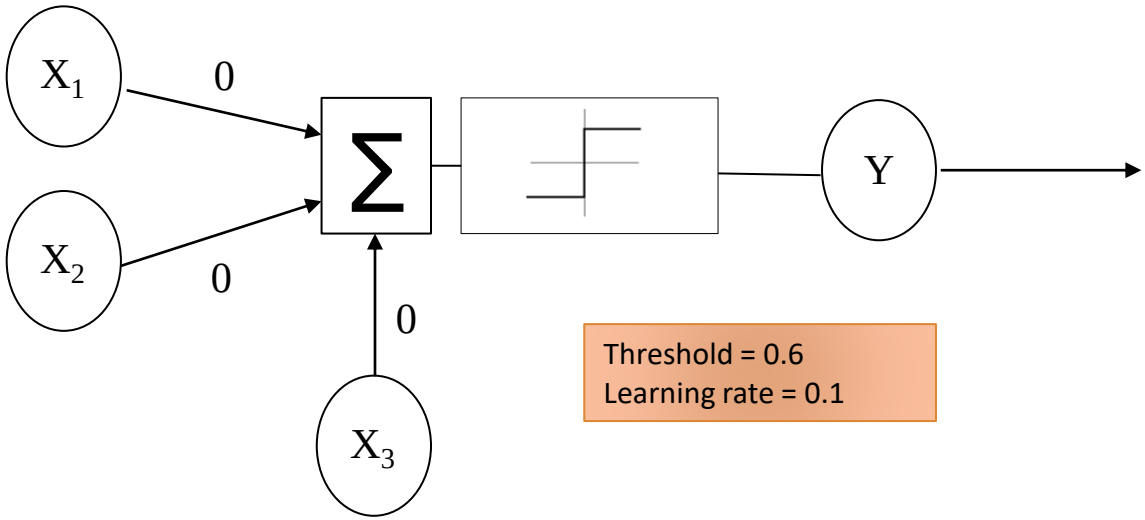


Threshold = 0.6
Learning rate = 0.1

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							

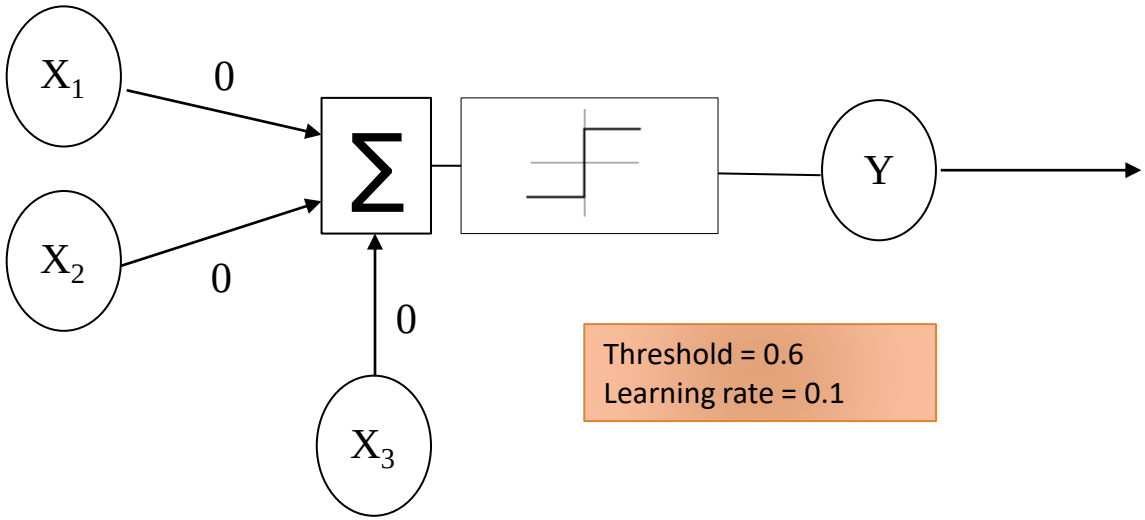


Threshold = 0.6
Learning rate = 0.1

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							

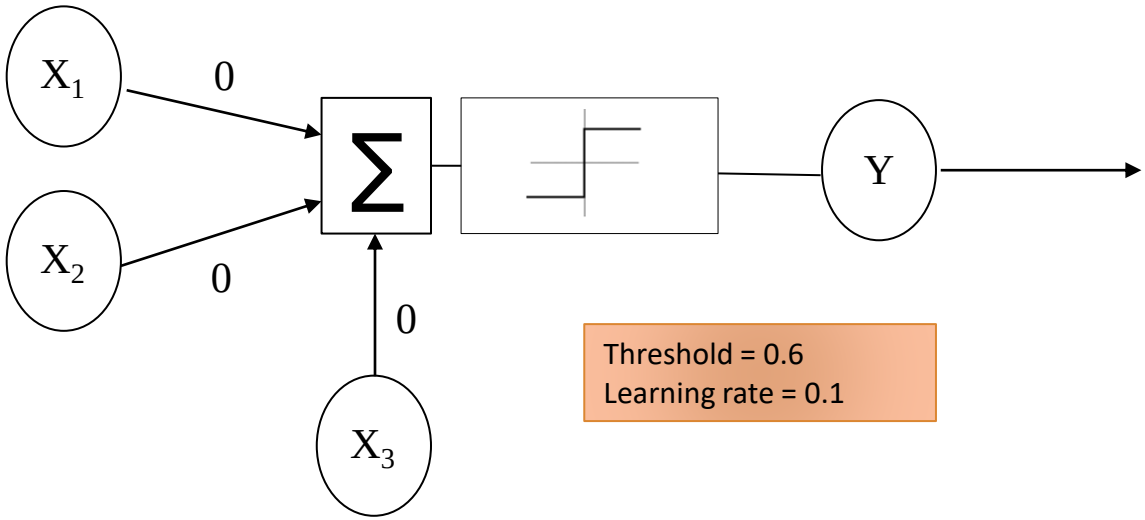


Threshold = 0.6
Learning rate = 0.1

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							

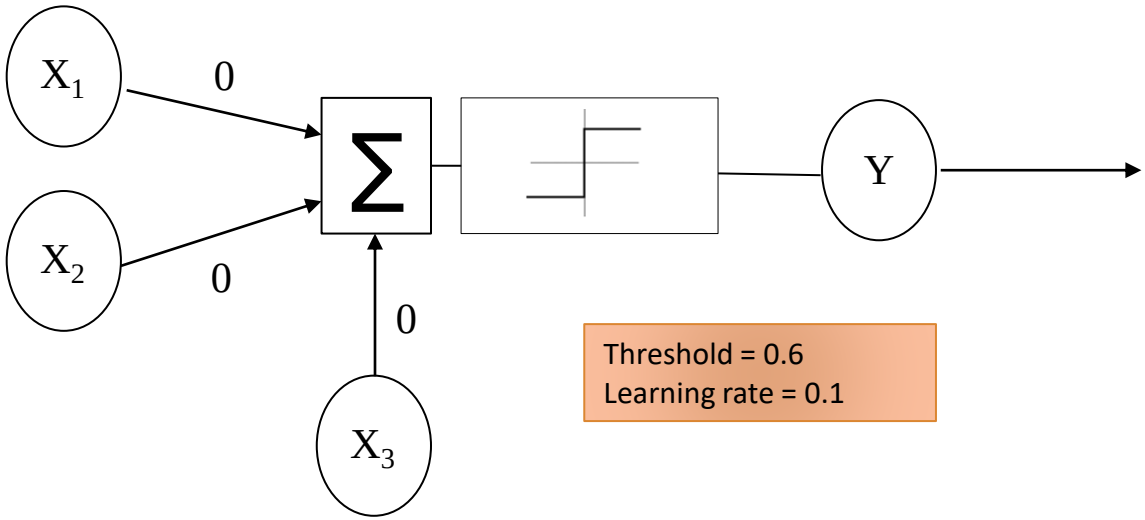


Threshold = 0.6
Learning rate = 0.1

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							

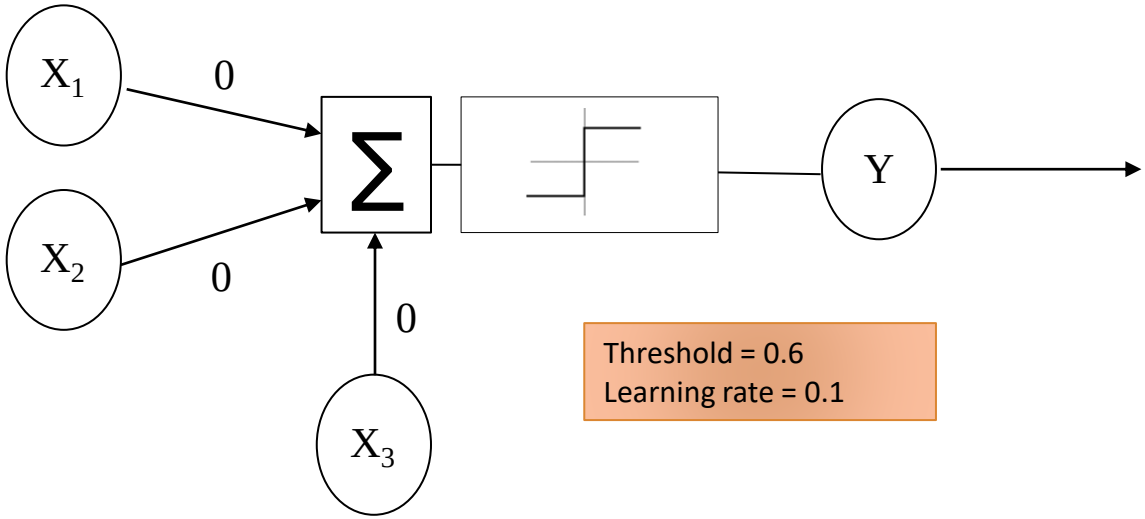


Threshold = 0.6
Learning rate = 0.1

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							

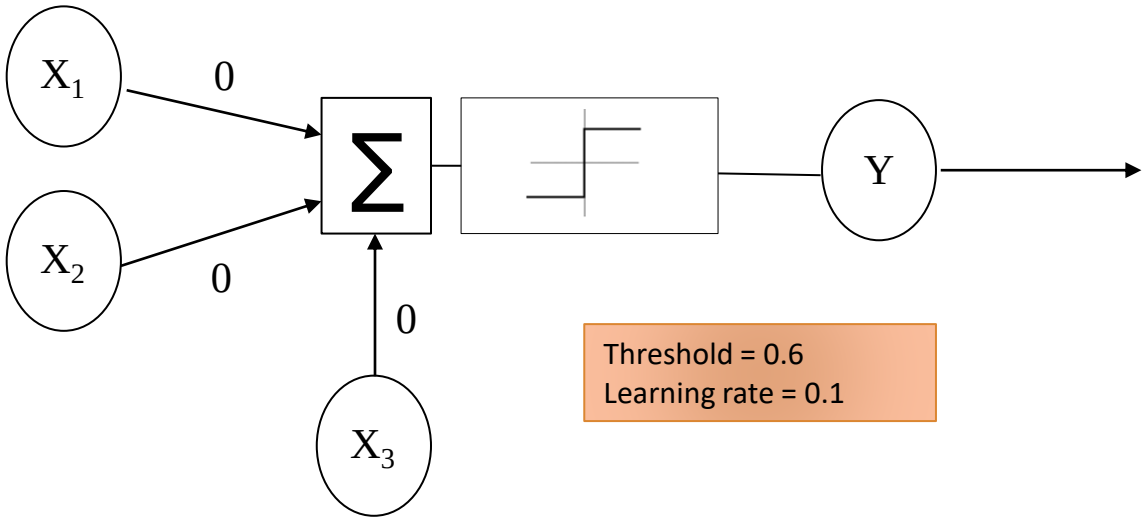


Threshold = 0.6
Learning rate = 0.1

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			

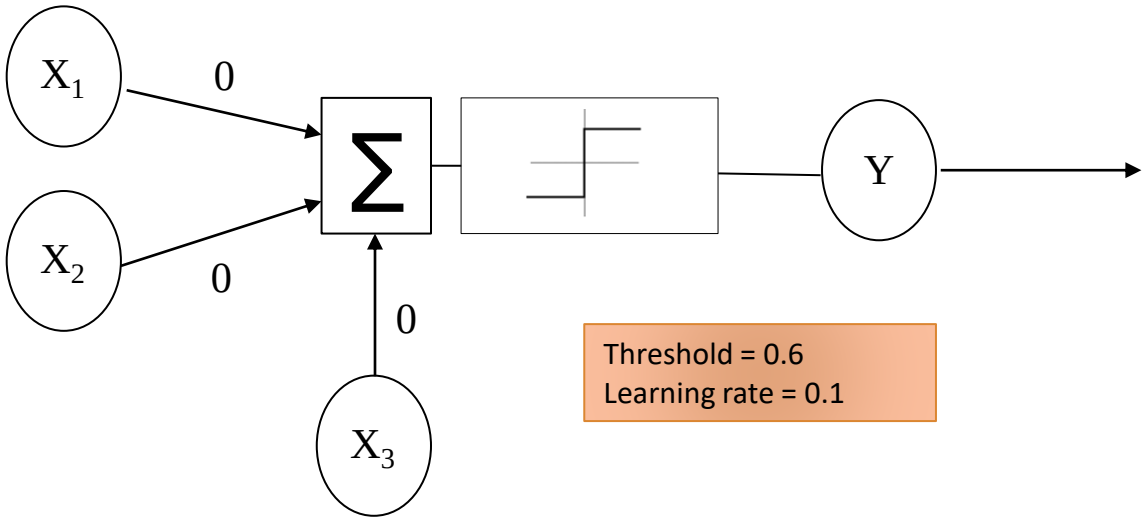


Threshold = 0.6
Learning rate = 0.1

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

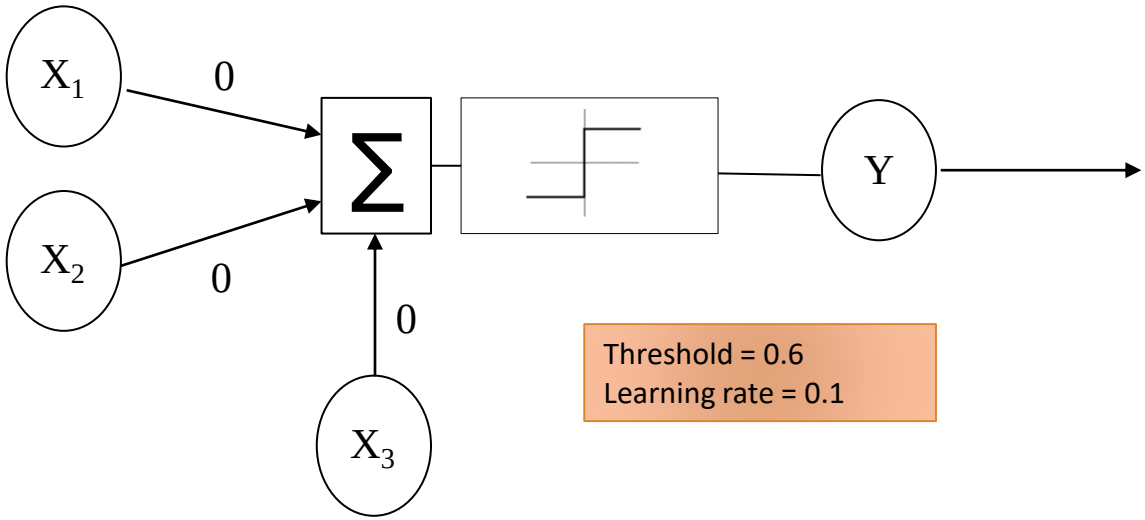


Threshold = 0.6
Learning rate = 0.1

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							

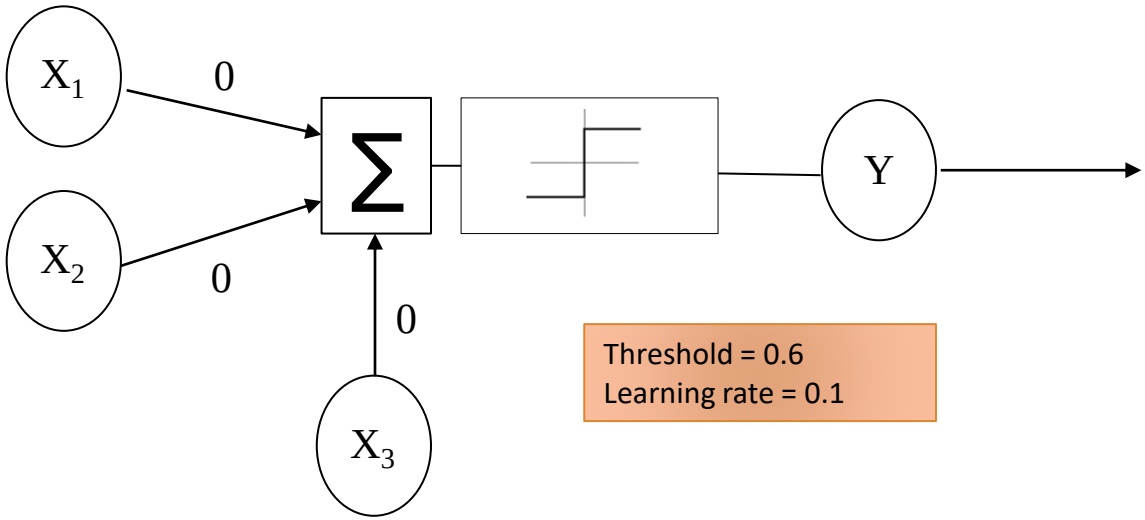


Threshold = 0.6
Learning rate = 0.1

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							

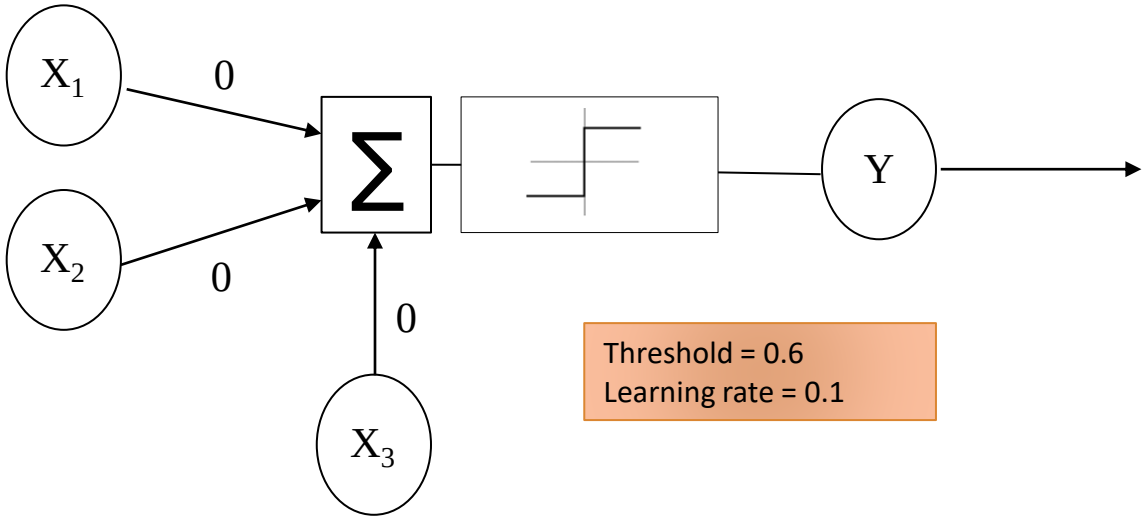


Threshold = 0.6
Learning rate = 0.1

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							

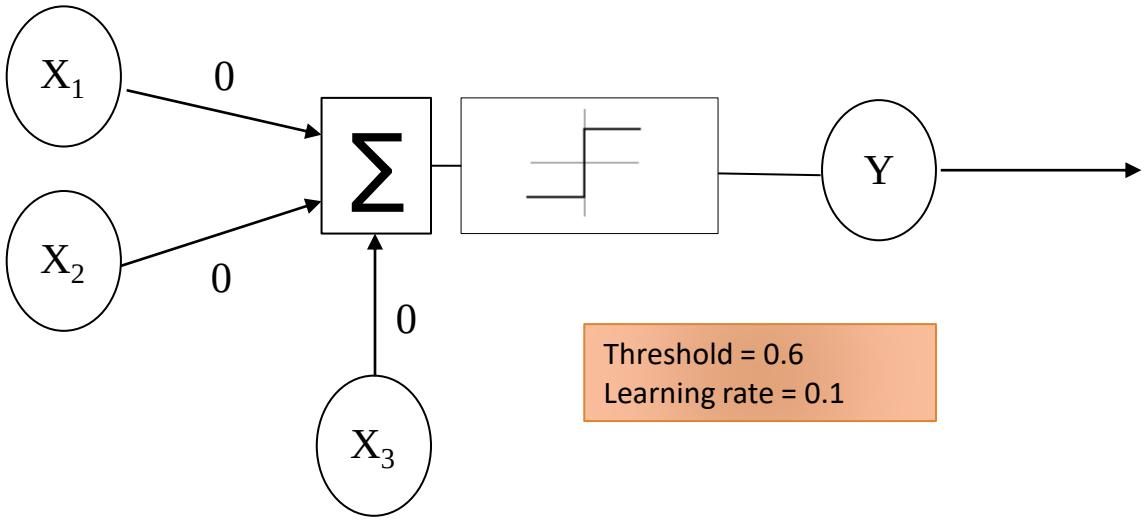


Threshold = 0.6
Learning rate = 0.1

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							

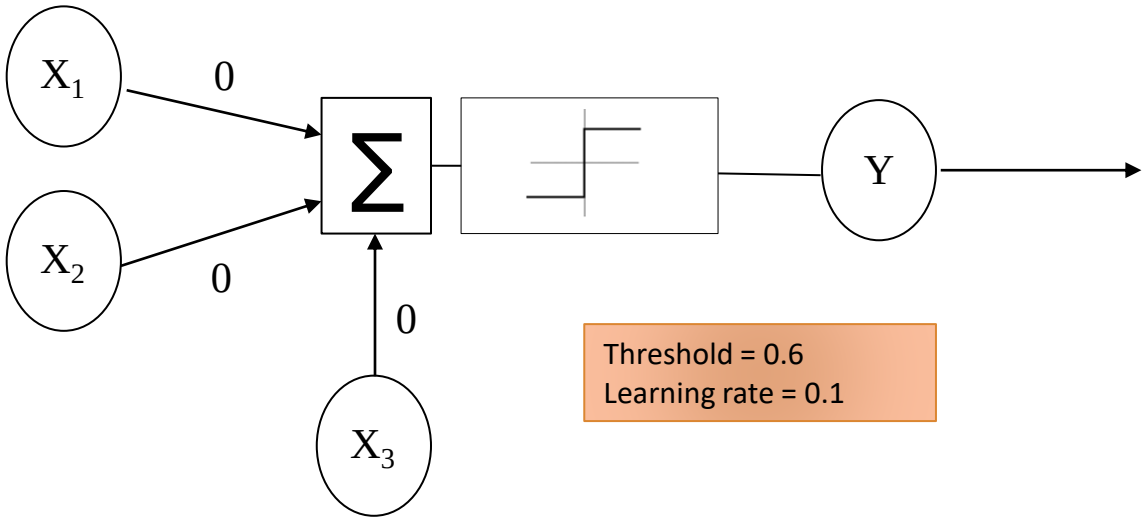


Threshold = 0.6
Learning rate = 0.1

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							

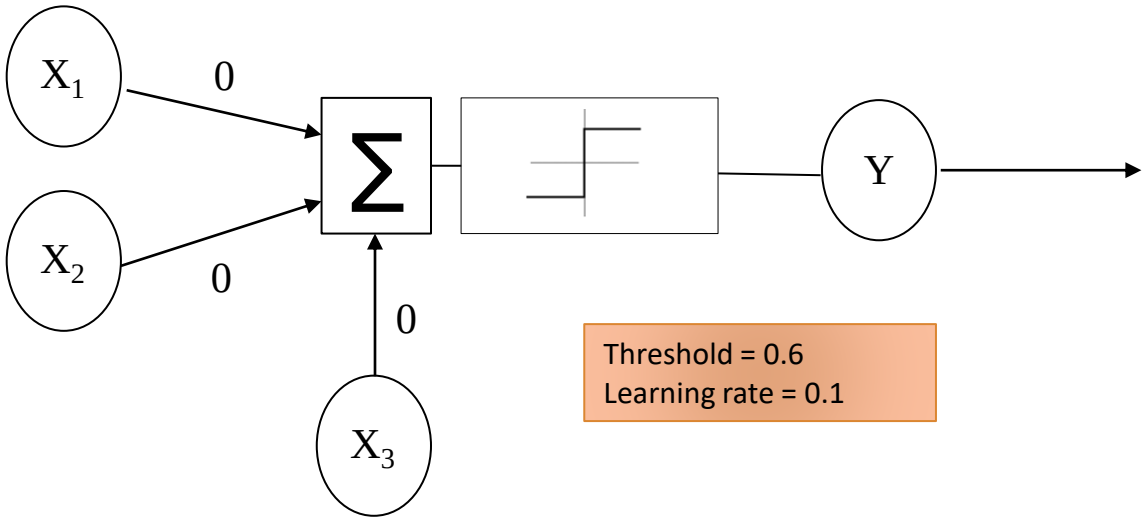


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

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

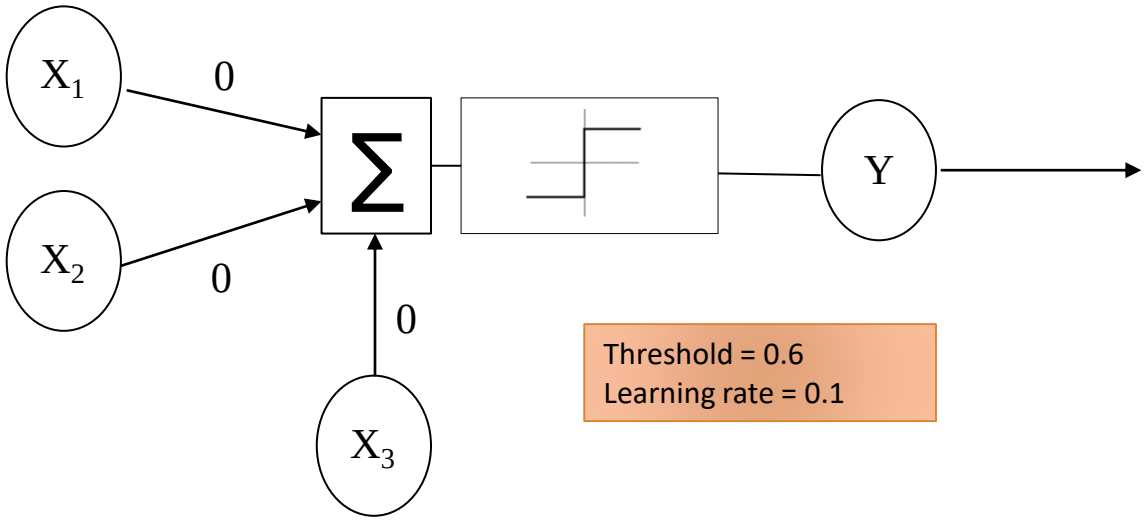
X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	0.0	0.1	0.4	0.4	0	1	+0.1
0	1	1	0.0	0.1	0.5	0.6	1	0	∓0.0
1	0	1	0.0	0.1	0.5	0.5	0	1	+0.1
1	1	1	0.1	0.1	0.6	0.8	1	-1	-0.1



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							

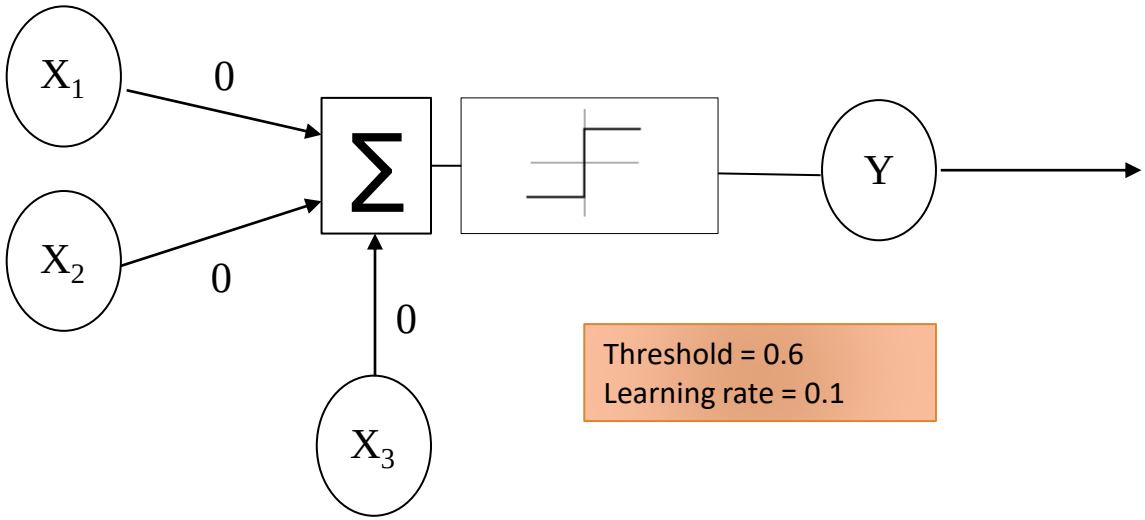


Threshold = 0.6
Learning rate = 0.1

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							

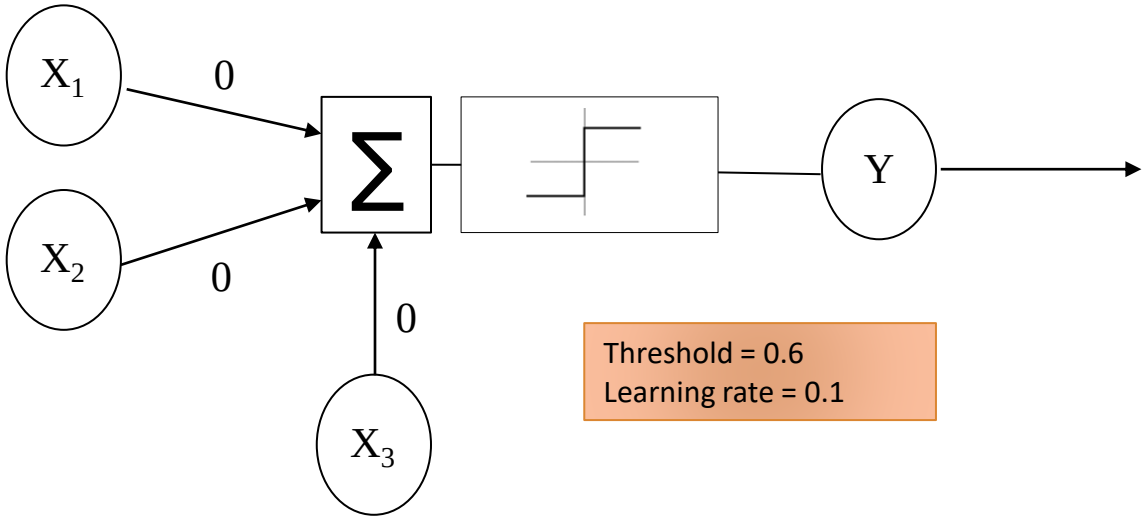


Threshold = 0.6
Learning rate = 0.1

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							

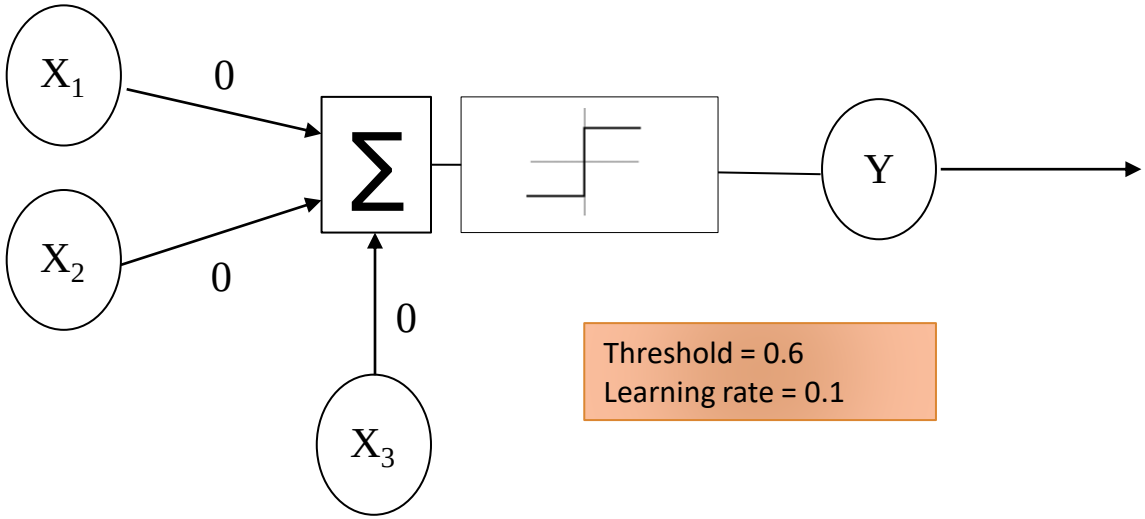


Threshold = 0.6
Learning rate = 0.1

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

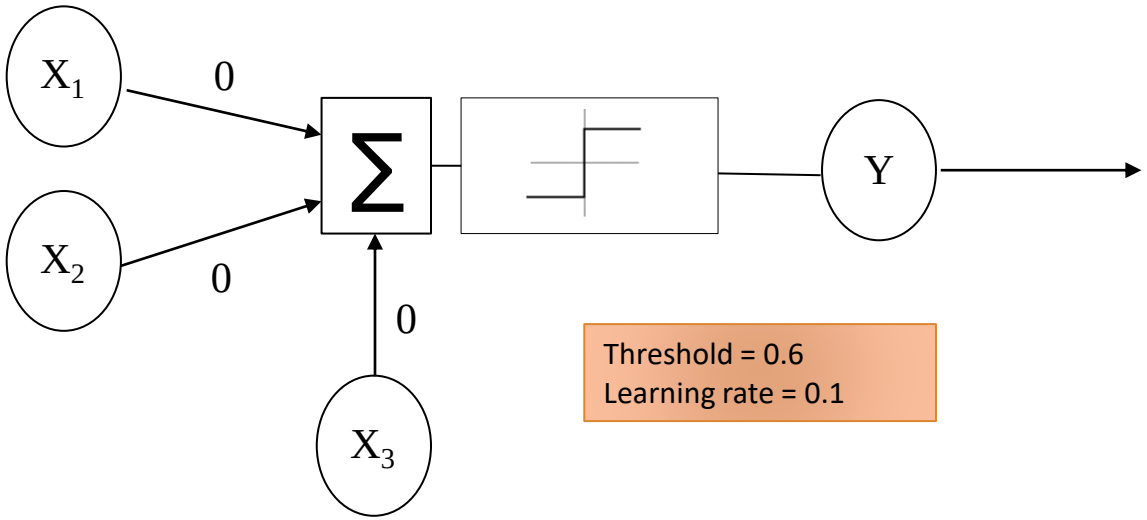


Threshold = 0.6
Learning rate = 0.1

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							

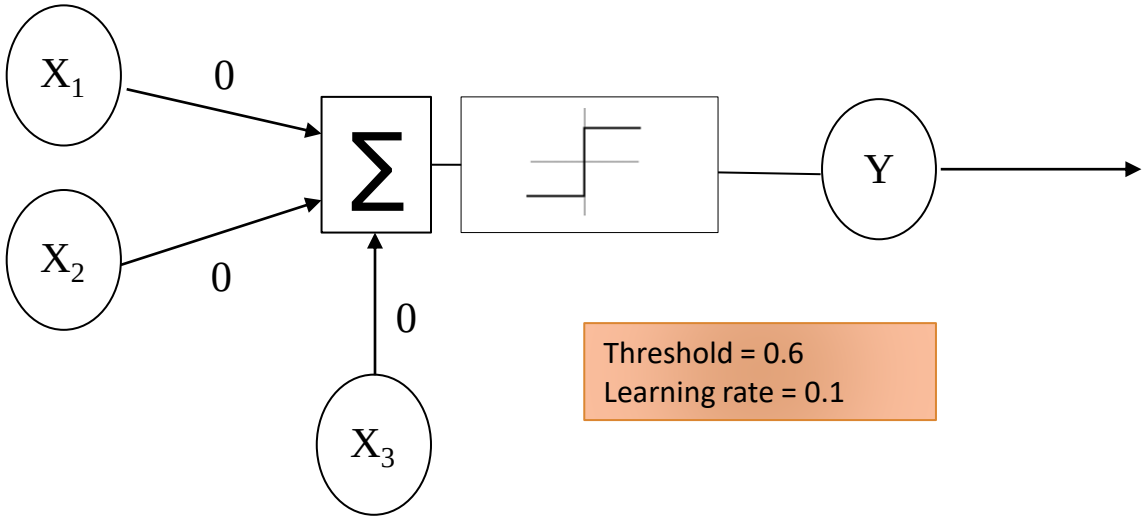


Threshold = 0.6
Learning rate = 0.1

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							

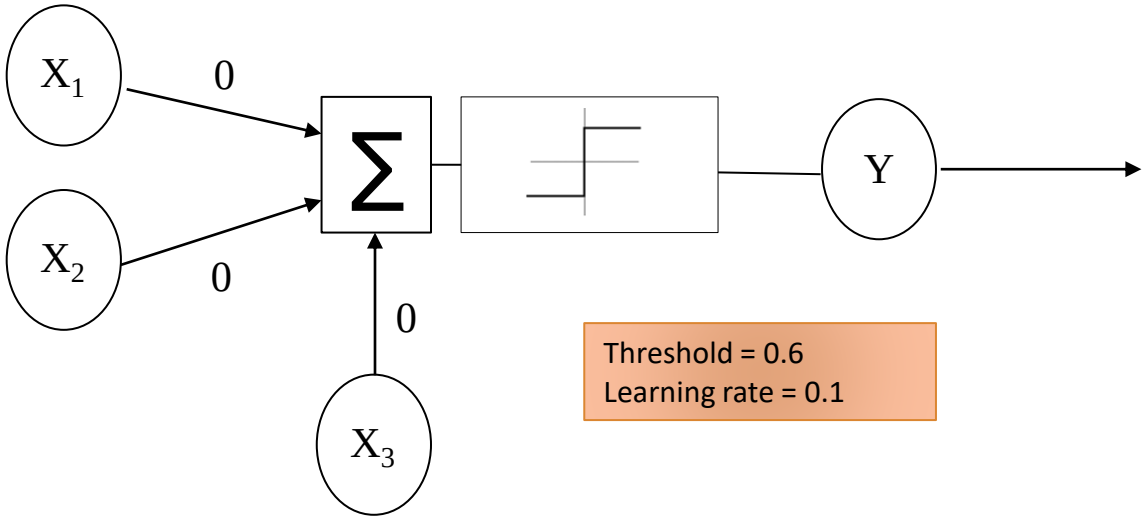


Threshold = 0.6
Learning rate = 0.1

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							

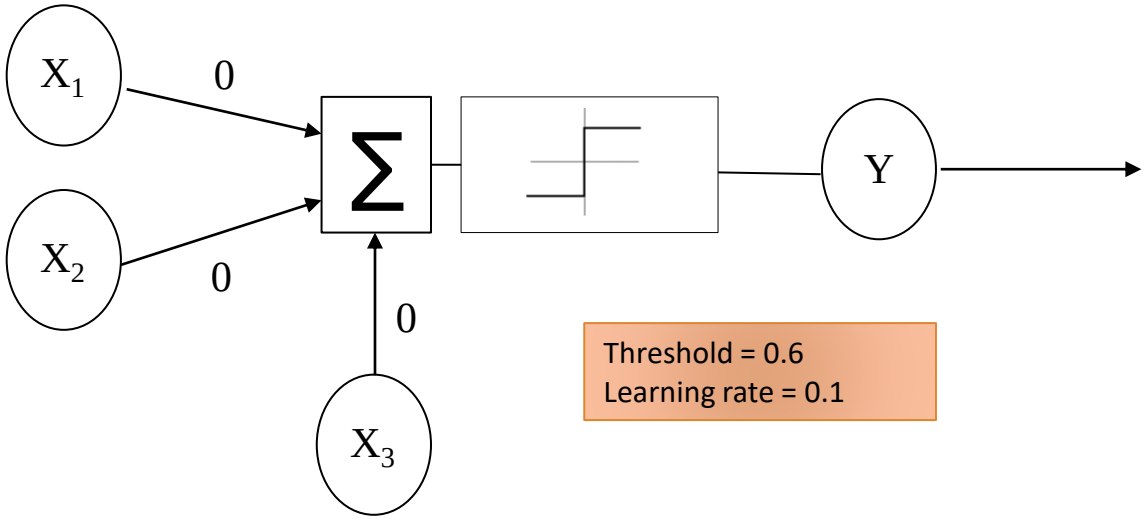


Threshold = 0.6
Learning rate = 0.1

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

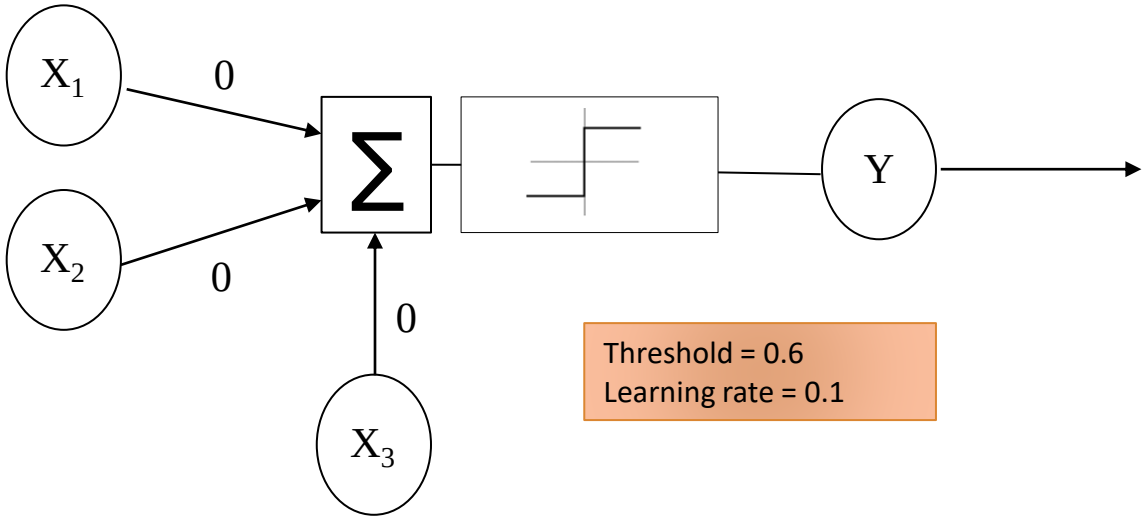


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

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

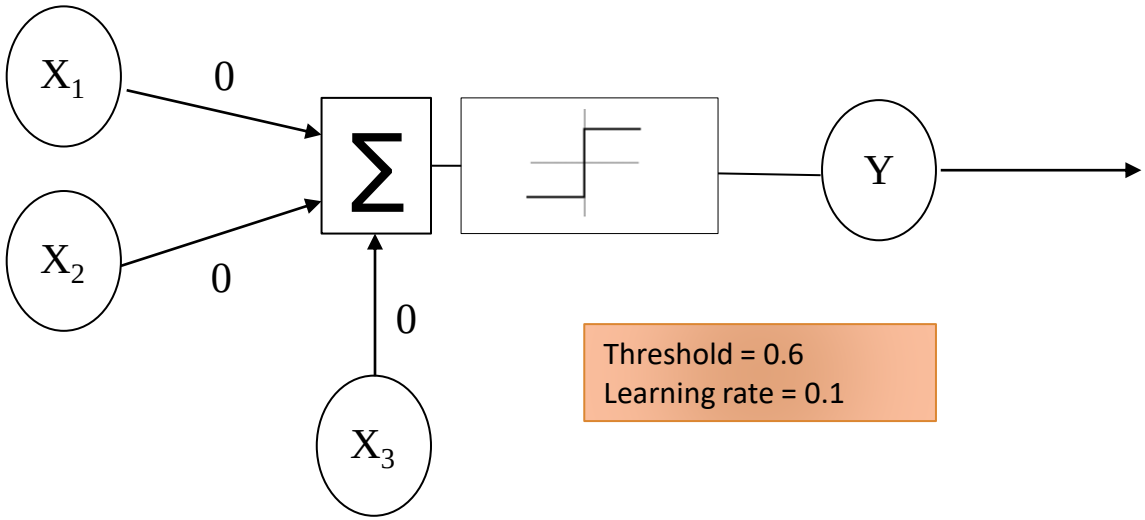
X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.2	-0.1	0.6	0.6	1	0	$\mp$0.0
0	1	1	-0.2	-0.1	0.6	0.5	0	1	+0.1
1	0	1	-0.2	0.0	0.7	0.5	0	1	+0.1
1	1	1	-0.1	0.0	0.8	0.7	1	-1	-0.1



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							

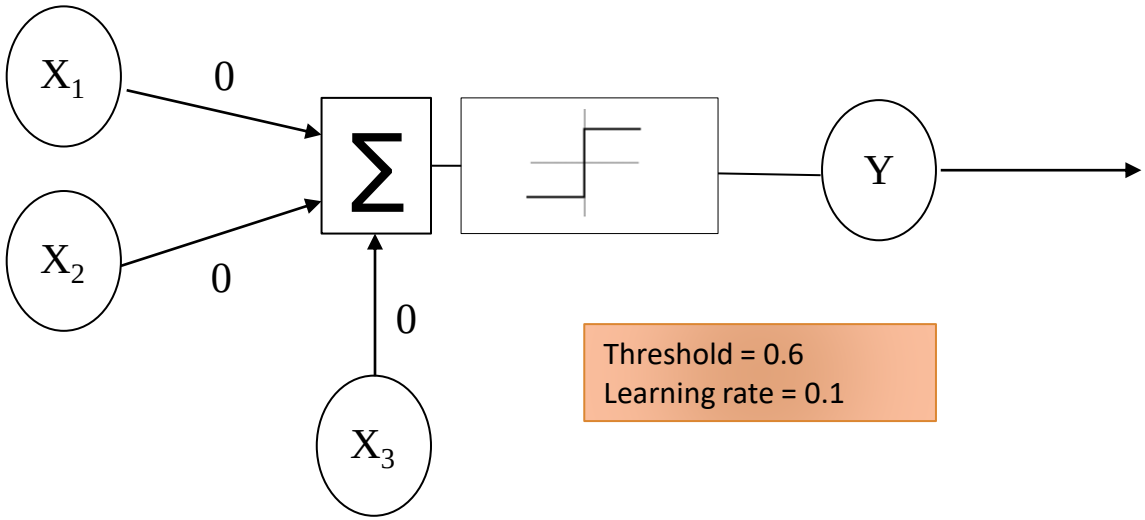


Threshold = 0.6
Learning rate = 0.1

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							

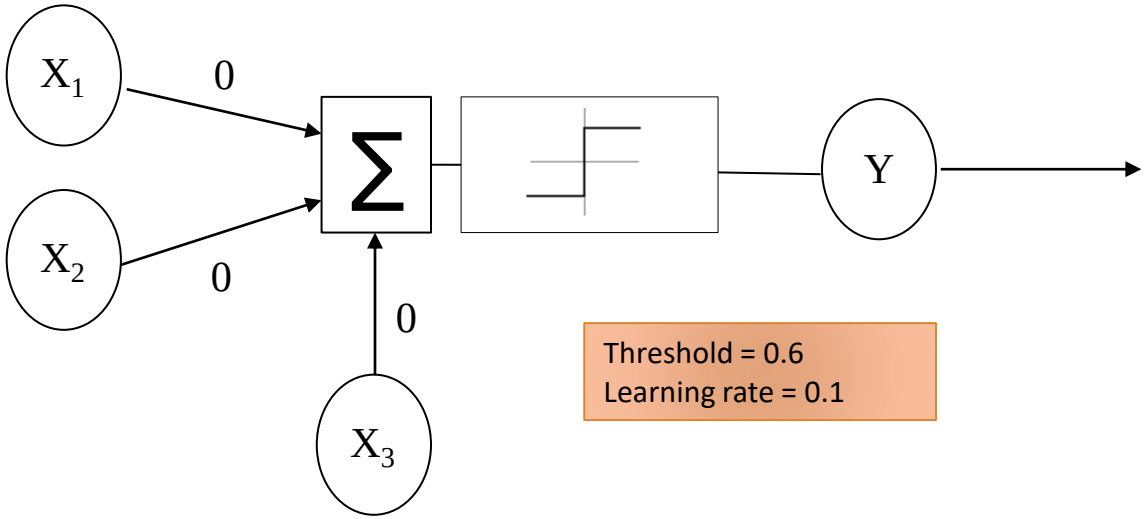


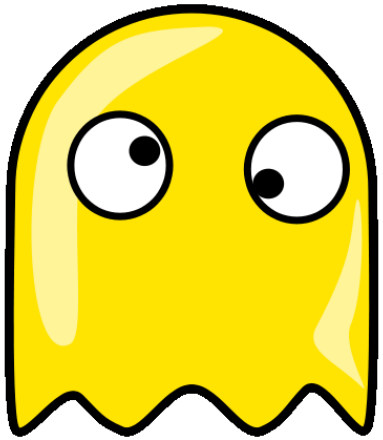
Threshold = 0.6
Learning rate = 0.1

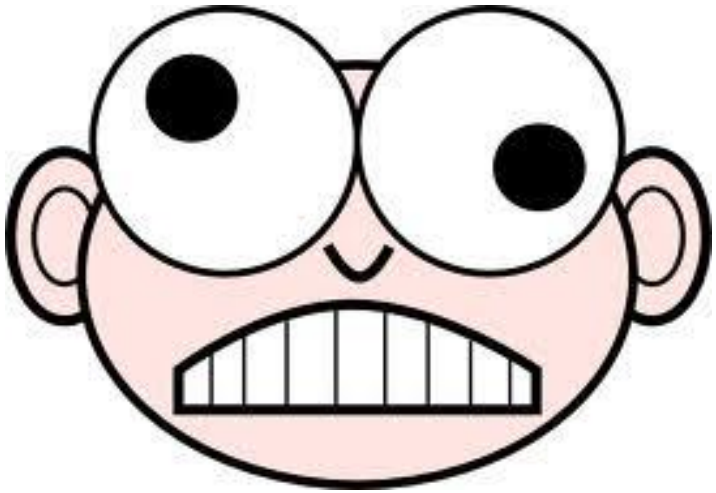
TRAINING A PERCEPTRON

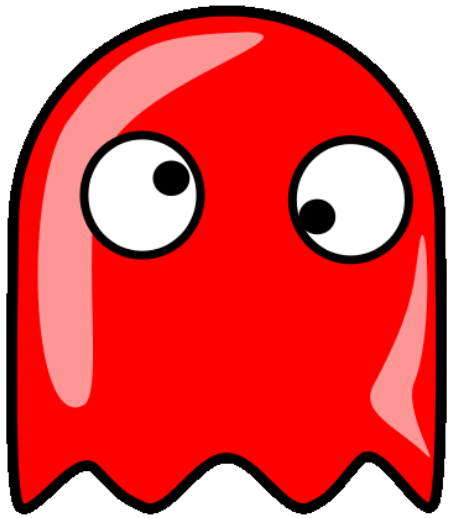
Input 1	Input 2	Desired Output
0	0	1
0	1	1
1	0	1
1	1	0

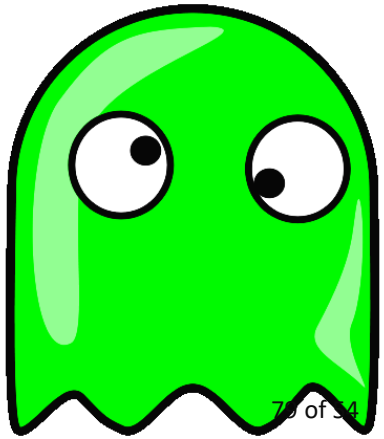
X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.2	-0.1	0.7	0.7	1	0	$\mp$0.0
0	1	1	-0.2	-0.1	0.7	0.6	1	0	$\mp$0.0
1	0	1	-0.2	-0.1	0.7	0.5	0	1	+0.1
1	1	1	-0.1	-0.1	0.8	0.6	1	-1	-0.1







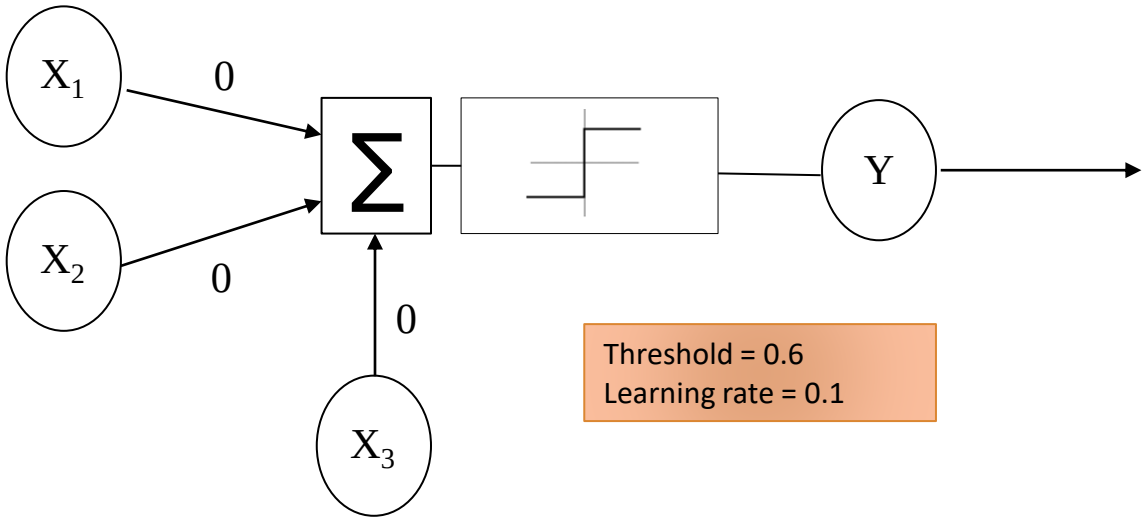




TRAINING A PERCEPTRON

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							

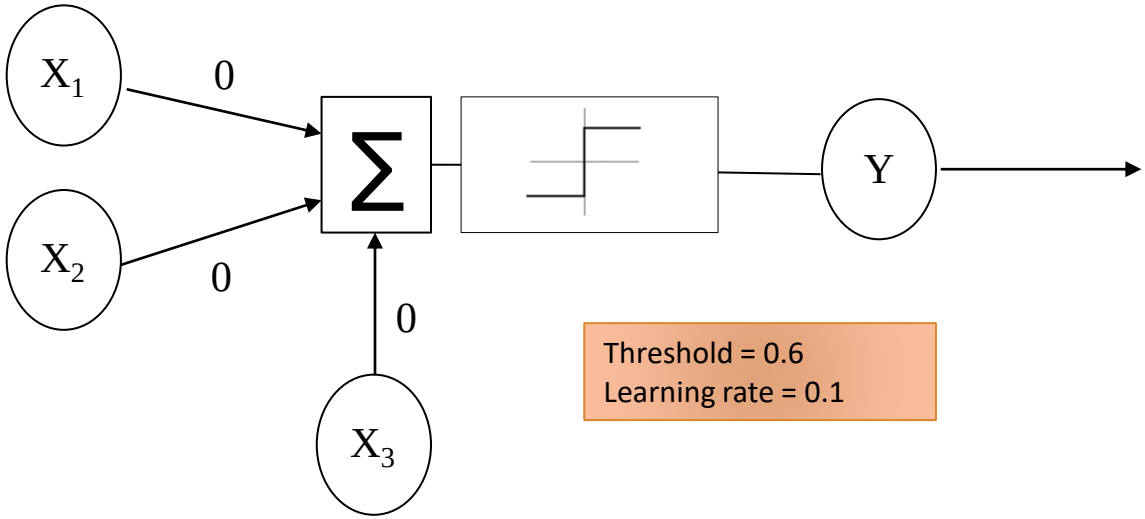


Threshold = 0.6
Learning rate = 0.1

TRAINING A PERCEPTRON

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

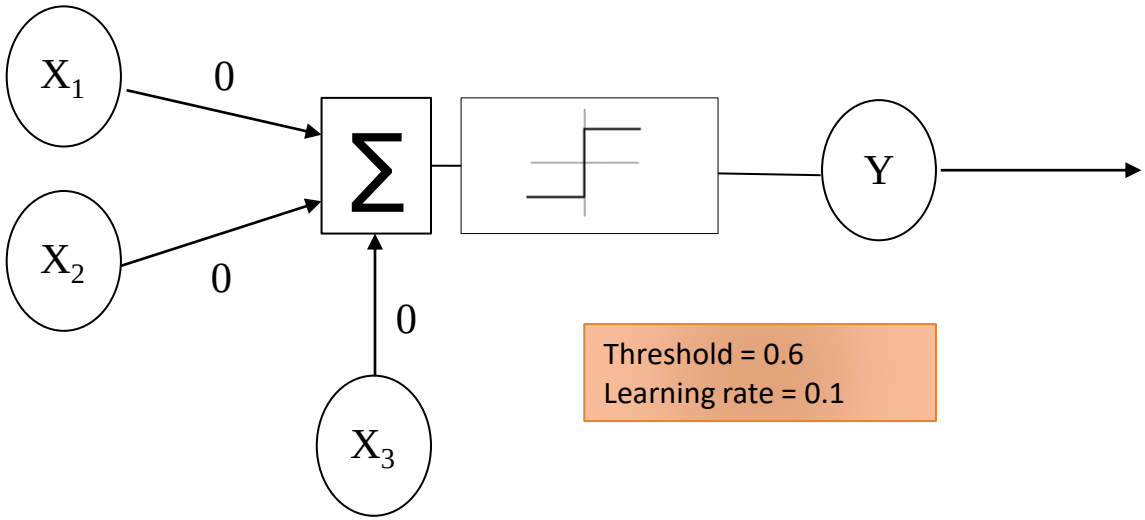
X_1	X_2	X_3	W_1	W_2	W_3	Sum	Output	Error	Correction
0	0	1	-0.2	-0.2	0.7	0.7	1	0	$\mp$0.0
0	1	1	-0.2	-0.2	0.7	0.5	0	1	+0.1
1	0	1	-0.2	-0.1	0.8	0.6	1	0	$\mp$0.0
1	1	1	-0.2	-0.1	0.8	0.5	0	0	$\mp$0.0



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	$\mp$0.0
0	1	1							
1	0	1							
1	1	1							

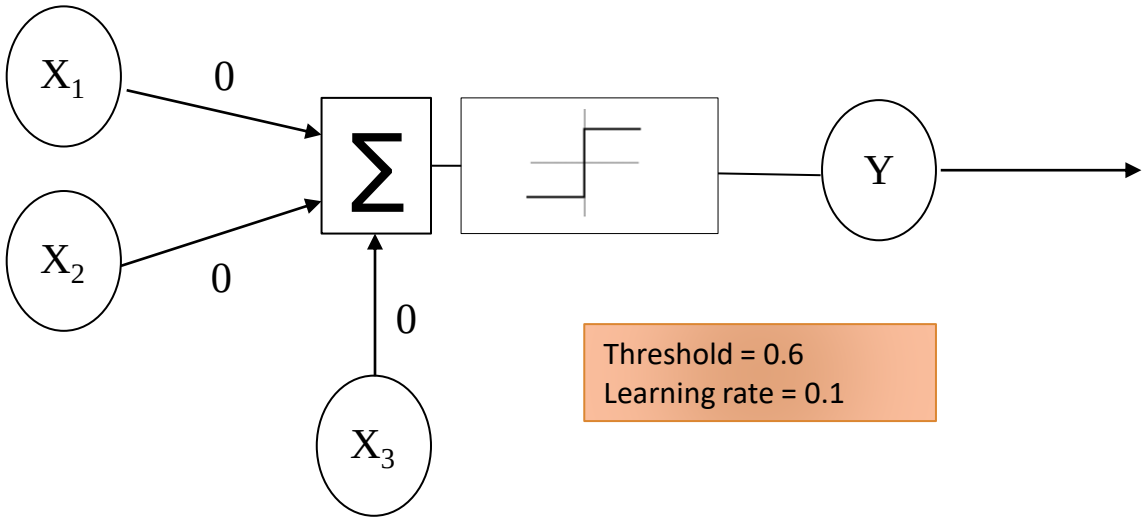


Threshold = 0.6
Learning rate = 0.1

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	$\mp$0.0
0	1	1	-0.2	-0.1	0.8	0.7	1	0	$\mp$0.0
1	0	1							
1	1	1							

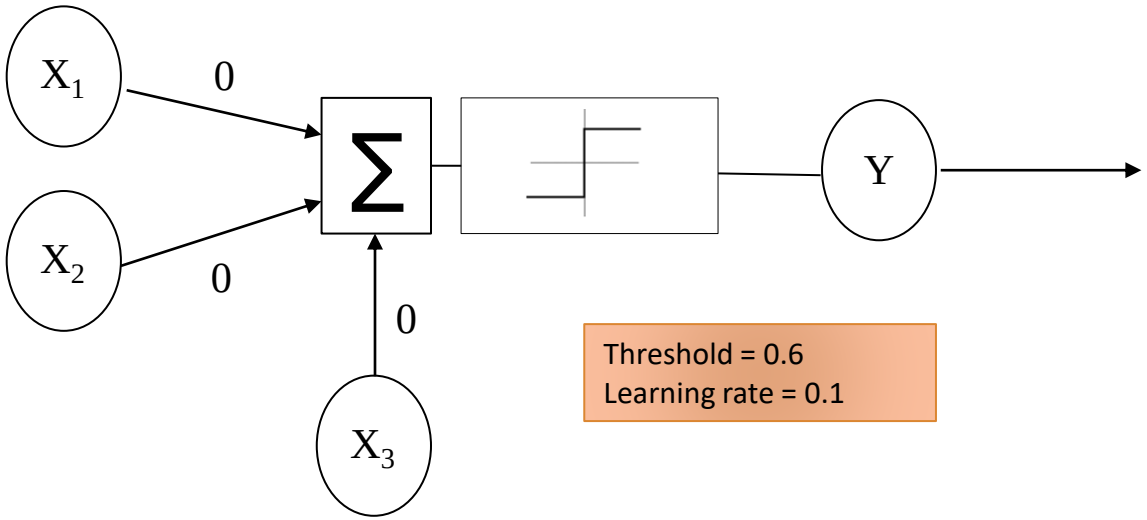


Threshold = 0.6
Learning rate = 0.1

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	0.8	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							

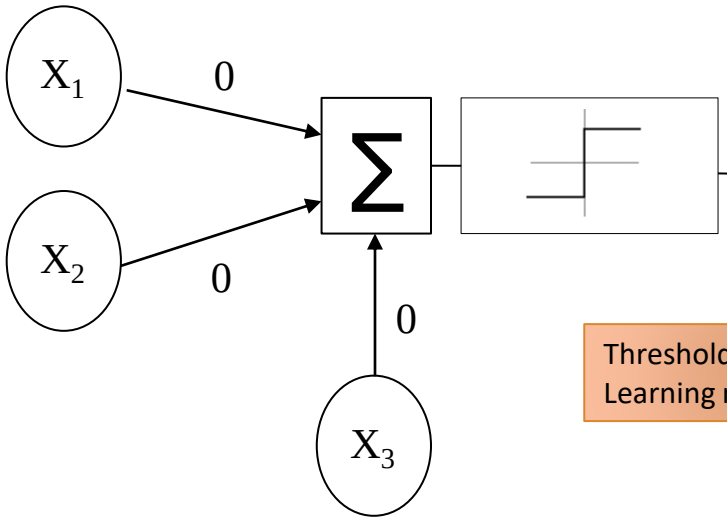


Threshold = 0.6
Learning rate = 0.1

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