# Perceptron

**Dr. P. K. Chaurasia**

Associate Professor,
Department of Computer Science and Information Technology
MGCUB, Motihari, Bihar

# Introduction

- A **perceptron** is a simple model of a biological neuron in an artificial neural network.
- The **perceptron** algorithm was designed to classify visual inputs, categorizing subjects into one of two types and separating groups with a line.
- Classification is an important part of machine learning and image processing.
- **Perceptron** was introduced by Frank Rosenblatt in 1957.

# cont…

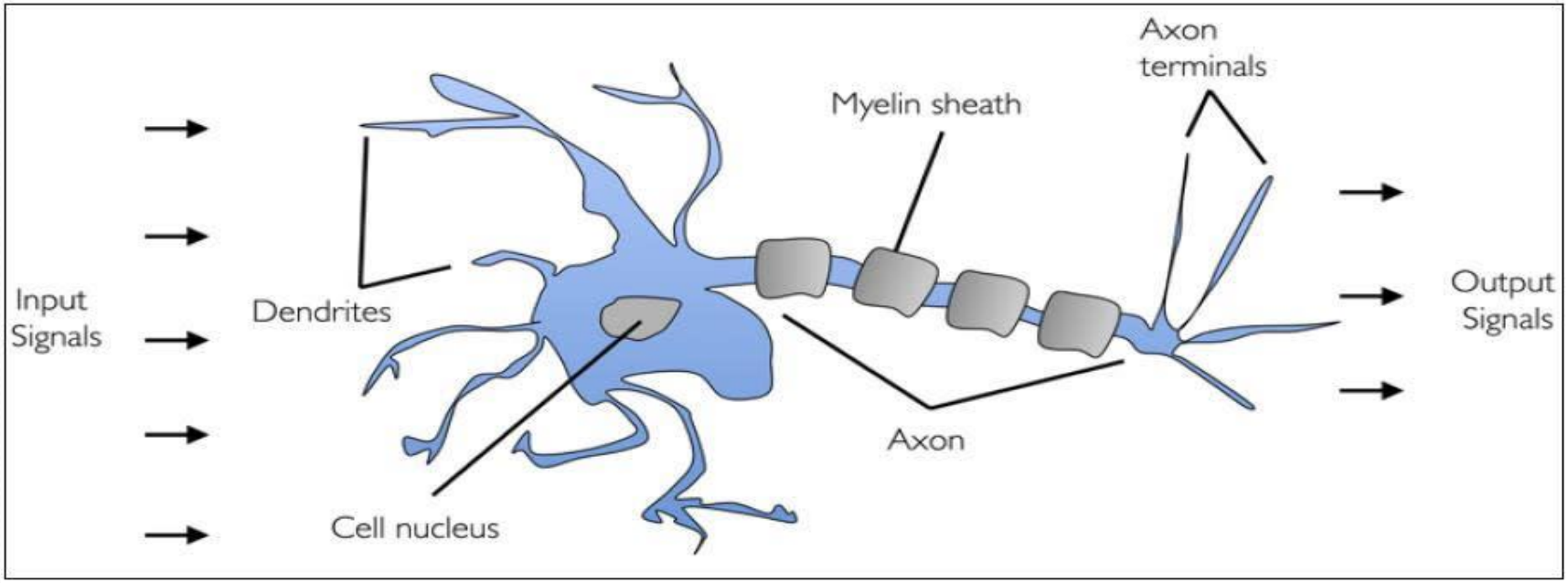
- Each **perceptron** sends multiple signals, one signal going to each **perceptron** in the next layer. For each signal, the **perceptron** uses **different** weights.
- One **difference between** an MLP and a **neural network** is that in the classic **perceptron**, the decision function is a step function and the output is binary.
- **Perceptron** is a single layer neural network and a multi-layer **perceptron** is called Neural Networks.
- **Perceptron** is a linear classifier (binary).
- It is used in supervised learning. It helps to classify the given input data.

# Biological Neurons

Cell nucleus or soma processes the information received from dendrites.

Axon is a cable that is used by neurons to send information.

Synapse is the connection between an axon and other neuron dendrites.

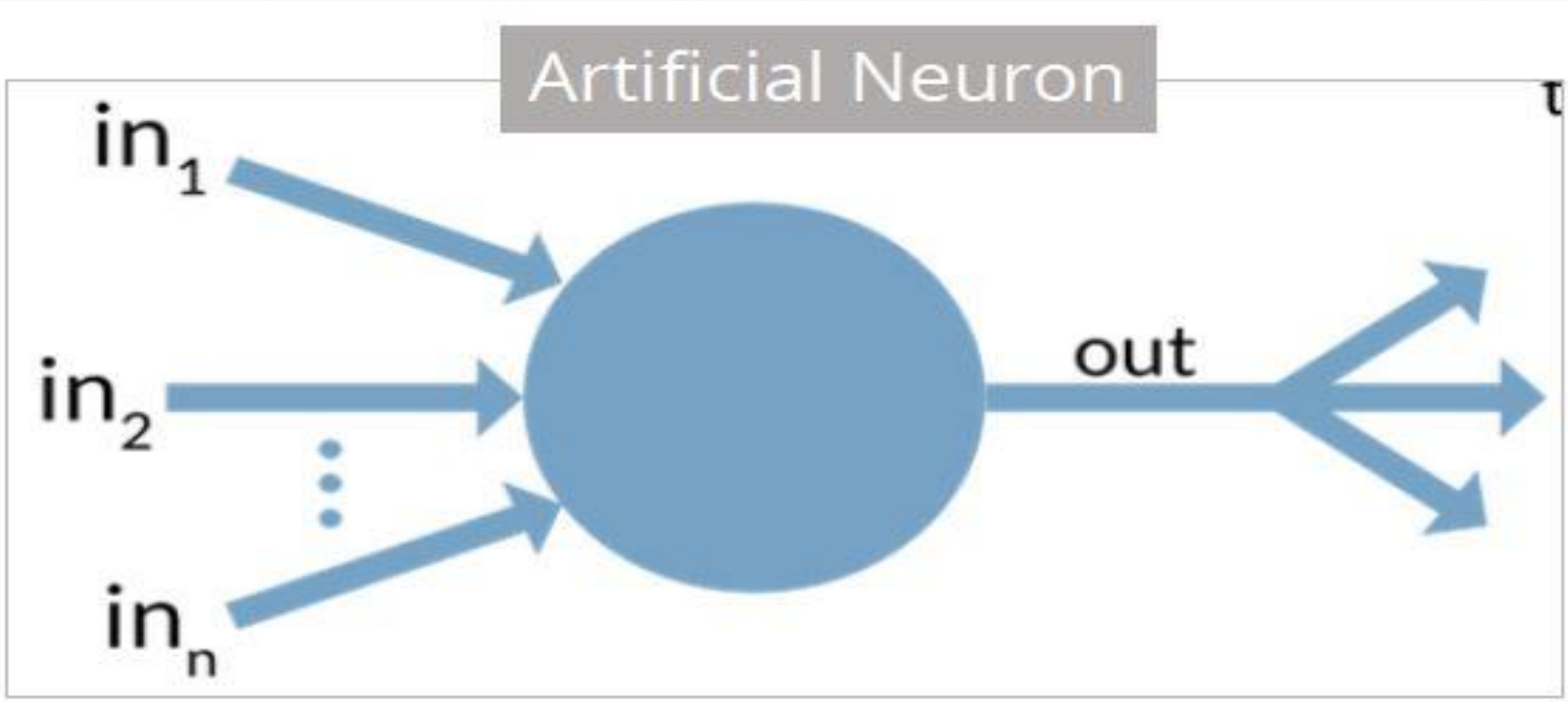


# Rise of Artificial Neurons

- Researchers Warren McCullock and Walter Pitts published their first concept of simplified brain cell in 1943.
- This was called McCullock-Pitts (MCP) neuron.
- They described such a nerve cell as a simple logic gate with binary outputs.
- Multiple signals arrive at the dendrites and are then integrated into the cell body.
- If the accumulated signal exceeds a certain threshold, an output signal is generated that will be passed on by the axon.

# Artificial Neurons

- An artificial neuron is a mathematical function based on a model of biological neurons, where each neuron takes inputs, weighs them separately, sums them up and passes this sum through a nonlinear function to produce output.



# Artificial Neurons

- It has the following characteristics:
  - A neuron is a mathematical function modeled on the working of biological neurons.
  - It is an elementary unit in an artificial neural network.
  - One or more inputs are separately weighted.
  - Inputs are summed and passed through a nonlinear function to produce output.
  - Every neuron holds an internal state called activation signal.
  - Each connection link carries information about the input signal.
  - Every neuron is connected to another neuron via connection link.

# Biological Neurons vs. Artificial Neurons

- The biological neuron is analogous to artificial neurons in the following terms:

| Biological Neurons | Artificial Neurons |
|---|---|
| Cell Nucleus (Soma) | Node |
| Dendrites | Input |
| Synapse | Weights or interconnections |
| Axon | Output |

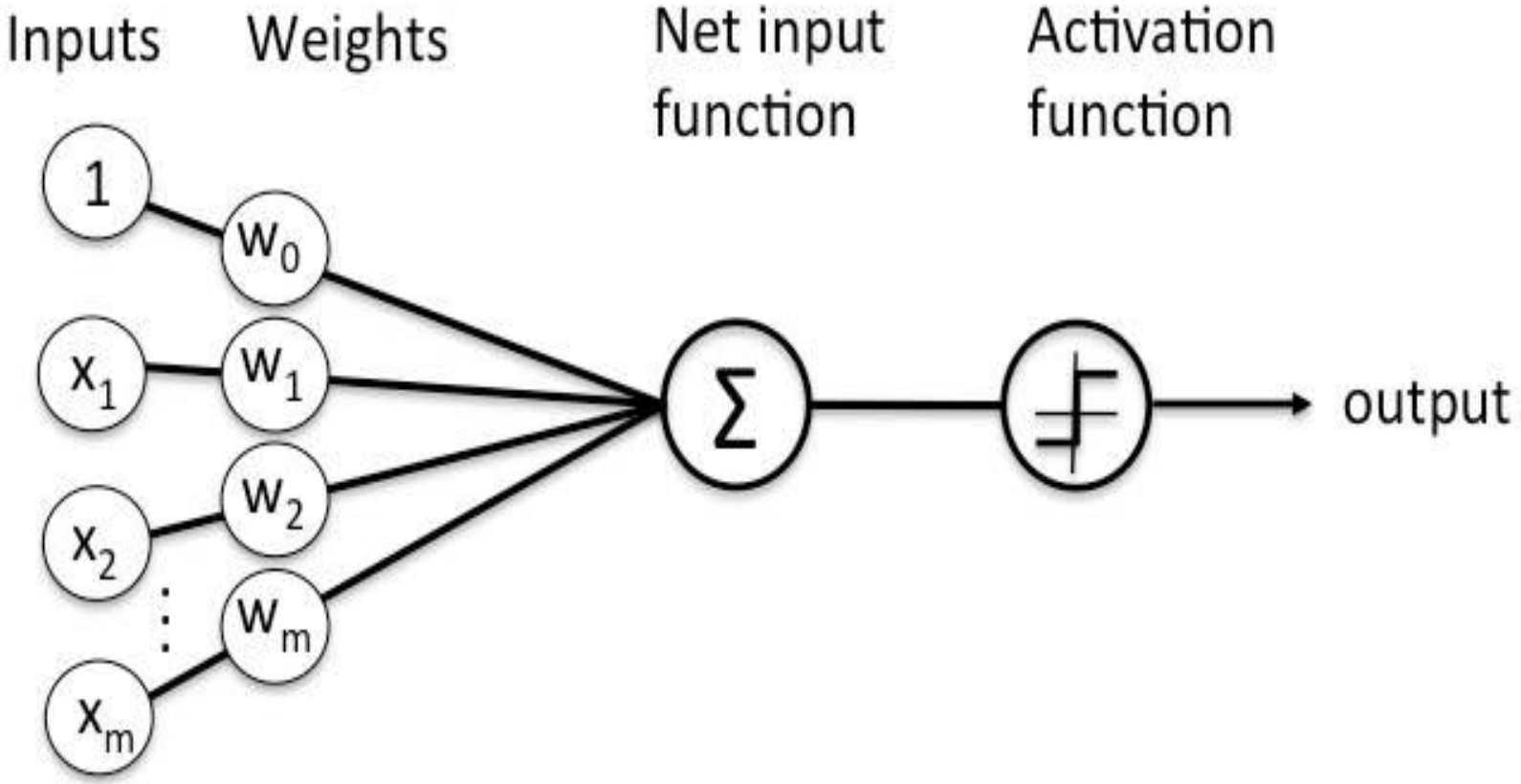
# Training Neural Network

- The neural network is a simple Perceptron, or a much more complicated multi-layer network with special activation functions.
- We need to develop a systematic procedure for determining appropriate connection weights.
- The network ***learn*** the appropriate weights from a representative set of training data.
- The simplest cases, however, direct computation of the weights are unmanageable.
- A good process is to start off with ***random initial weights*** and adjust them in small steps until the required outputs are generated.

# Perceptron

- A perceptron is a neural network unit that does certain computations to detect features or business intelligence in the input data.
- A Perceptron is an algorithm for supervised learning of binary classifiers.
- This algorithm enables neurons to learn and processes elements in the training set one at a time.
- A single artificial neuron that computes its weighted input and uses a threshold activation function.

# Perceptron


Inputs
Weights
Net input function
Activation function
1
$x_1$
$x_2$
$x_m$
$w_0$
$w_1$
$w_2$
$w_m$
Σ
output

# Types of Perceptron

- There are two types of perceptrons:
  - Single layer
  - Multilayer.
- **Single layer Perceptrons** can learn only linearly separable patterns.
- **Multilayer Perceptrons** or feedforward neural networks with two or more layers have the greater processing power.
- The Perceptron algorithm learns the weights for the input signals in order to draw a linear decision boundary.
- This enables you to distinguish between the two linearly separable classes +1 and -1.

# Perceptron Learning

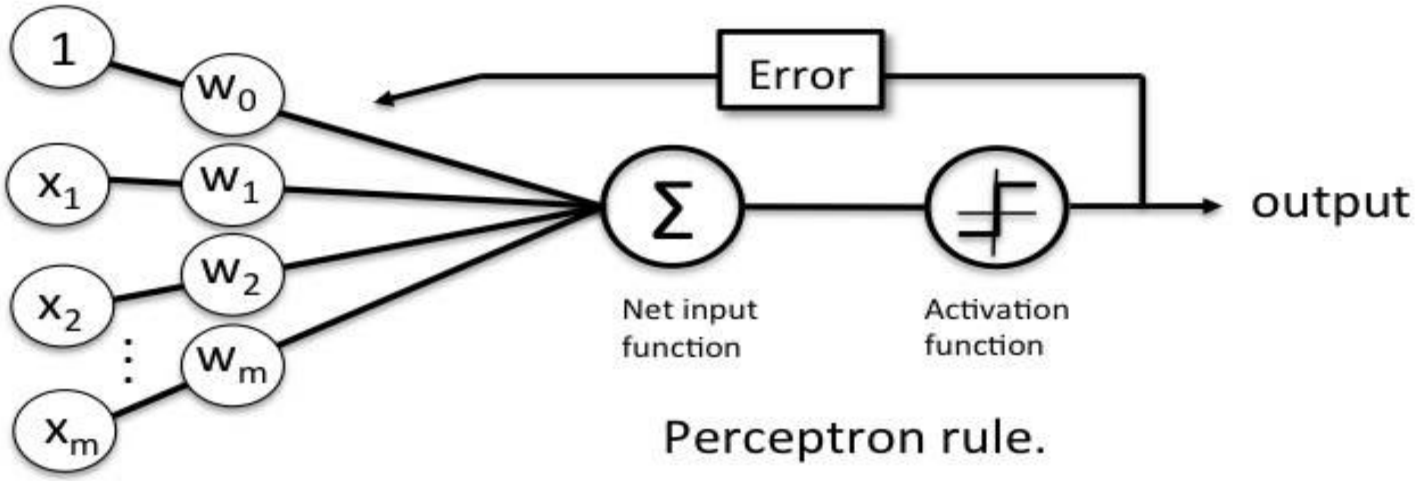
- The decision boundaries are hyperplanes, ***learning*** as the process of shifting around the hyperplanes, until each training pattern is classified correctly.
- Somehow, we need to formalise that process of "shifting around" into a systematic ***algorithm*** that can easily be implemented on a computer.
- The "shifting around" can be split up into a number of small steps.
- If the network weights at time $t$ are $w_{ij}(t)$ , then the shifting process corresponds to moving them by a small amount $\otimes w_{ij}(t)$ so that at time $t+1$ we have weights.

$$w_{ij}(t+1) = w_{ij}(t) + \otimes w_{ij}(t)$$

- It is convenient to treat the thresholds as weights, as discussed previously, so we don't need separate equations for them..

# Perceptron Learning Rule

- It states that the algorithm would automatically learn the optimal weight coefficients.
- The input features are then multiplied with these weights to determine if a neuron fires or not.
- The Perceptron receives multiple input signals, and if the sum of the input signals exceeds a certain threshold, it either outputs a signal or does not return an output.
- In the context of supervised learning and classification, this can then be used to predict the class of a sample.




Perceptron rule.

# Perceptron Function

- It is a function that maps its input "x," which is multiplied with the learned weight coefficient; an output value "f(x)"is generated.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

where

w = vector of real-valued weights.

b = bias (an element that adjusts the boundary away from origin without any dependence on the input value)

x = vector of input x values.

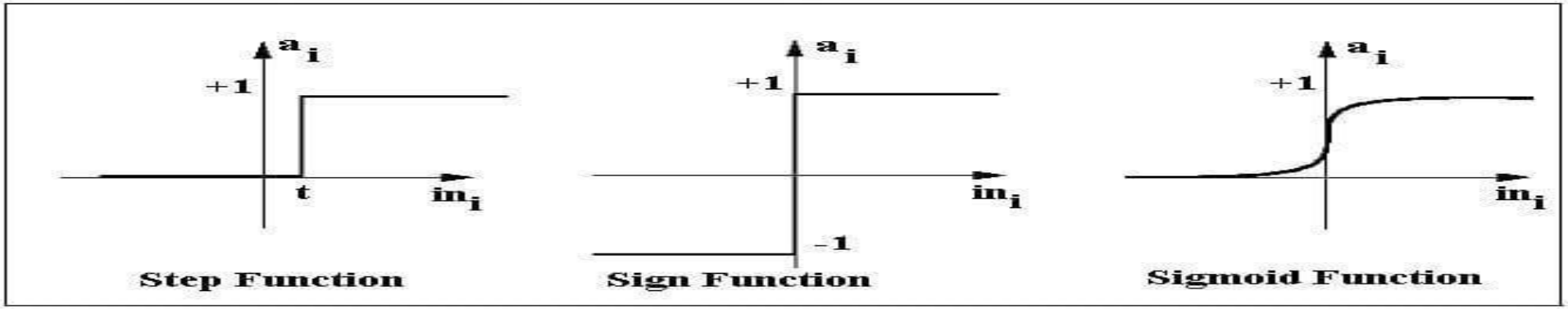
$$\sum_{i=1}^{m} w_i x_i$$

m= number of inputs to the perceptron.

The output can be represented as "1" or "0." It can also be represented as "1" or "-1" depending on which activation function is used.

# Activation Functions of Perceptron

- The activation function applies a step rule (convert the numerical output into +1 or -1) to check if the output of the weighting function is greater than zero or not.
- Function gets triggered above a certain value of the neuron output; else it outputs zero. Sign Function outputs +1 or -1 depending on whether neuron output is greater than zero or not. Sigmoid is the S-curve and outputs a value between 0 and 1.
- **For example:**
- If $\Sigma w_i x_i > 0$ then final output =1 else final output = - 1.



# Inputs of a Perceptron

- A Perceptron accepts inputs, moderates them with certain weight values, then applies the transformation function to output the final result.
- A Boolean output is based on inputs such as salaried, married, age, past credit profile, etc.
- It has only two values: Yes and No or True and False. The summation function "∑" multiplies all inputs of "x" by weights "w" and then adds them up as follows:

$$w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

# Output of Perceptron

- Perceptron with a Boolean output:
  - Input : X1…….Xn
  - Output: O(X1……Xn)

$$o(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0 \\ -1 \text{ otherwise} \end{cases}$$

- Weights Wi = contribution of input Xi to the perceptron output.
- Wo = bias or threshold.
- If $\sum w.x > 0$, output is +1, else -1. The neuron gets triggered only when weighted input reaches a certain threshold value.

$$o(\vec{x}) = sgn(\vec{w} \cdot \vec{x})$$

$$sgn(y) = \begin{cases} 1 \text{ if } y > 0 \\ -1 \text{ otherwise} \end{cases}$$

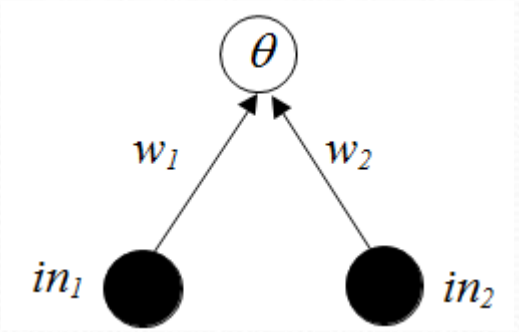
- An output of +1 specifies that the neuron is triggered. An output of -1 specifies that the neuron did not get triggered.
- “sgn” stands for sign function with output +1 or -1.

# Decision Boundary

- The weight vector is relevant to the decision boundary.
- The weight vector points in the direction of the vector which should produce an output of 1.
  - Vector with the positive output are on the right side of the decision boundary.
    - If w pointed in the opposite direction, the dot products of all input vectors would have the opposite sign.
    - Would result in same classification but opposite labels.
- The bias determines the position of the boundary.
  - $W^T P + b = 0$ using one point on the decision boundary to find b.

# Decision Boundary in Two Dimensions

- It is easy to visualise what the neural network is doing. It is forming decision boundaries between classes.
- Out = step ($w_1in_1 + w_2in_2 - \theta$)




- The decision boundary (between out = 0 and out =1)

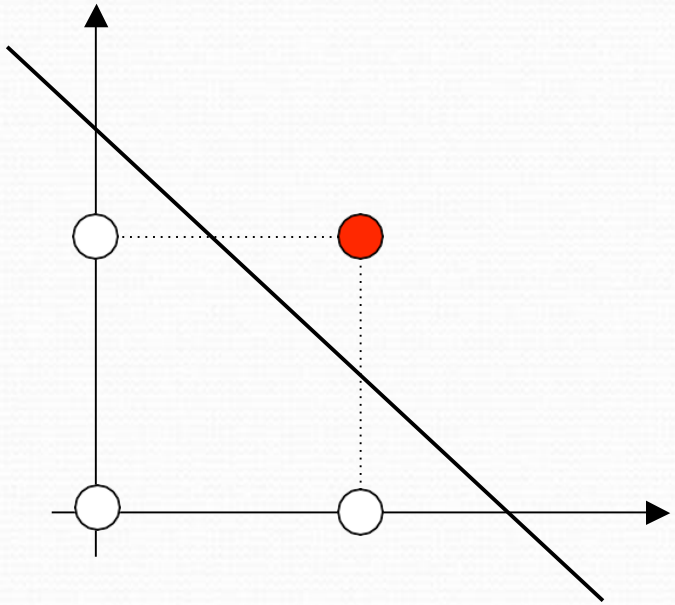
$$w_1in_1 + w_2in_2 - \theta = 0$$

- Therefore, in two dimensions the decision boundaries are always straight line.

# Decision Boundary for AND and OR

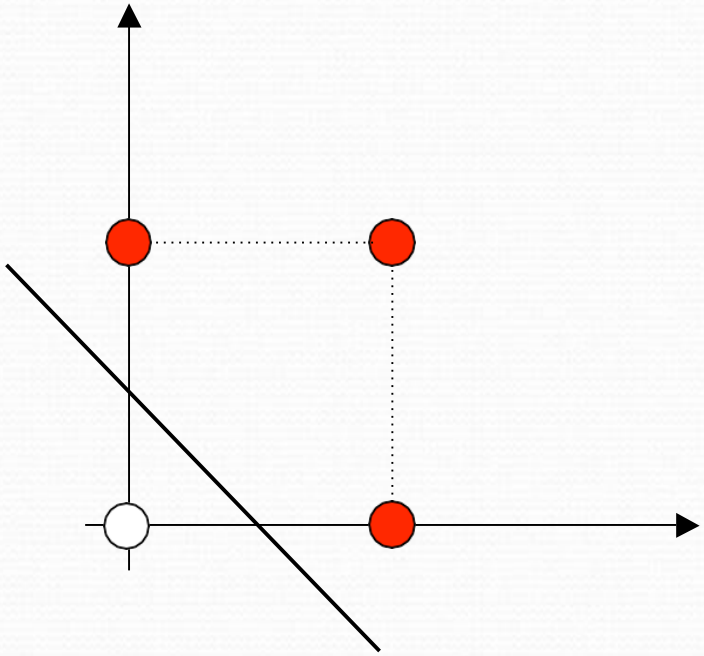
**AND**

$w_1 = 1, w_2 = 1, \theta = 1.5$



**OR**

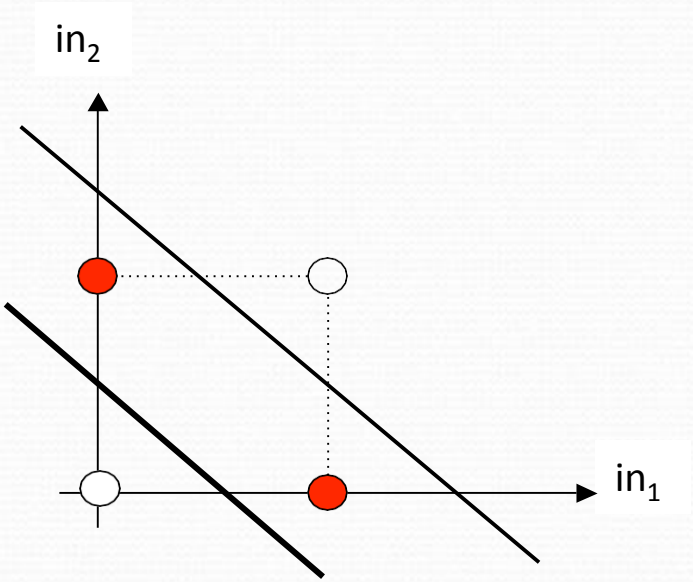
$w_1 = 1, w_2 = 1, q = 0.5$

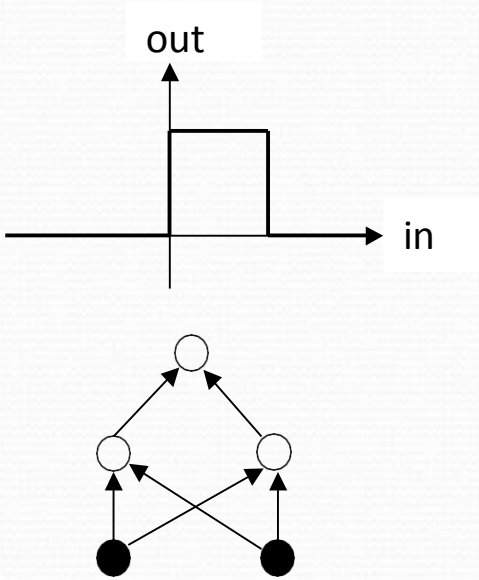


We can change the weights and thresholds without changing the output decisions

# Decision Boundaries for XOR

- We need two straight lines to separate the different outputs/decisions:




There are two examples:

- Either change the transfer function so that it has more than one decision boundary
- Use a more complex network that is able to generate more complex decision boundaries.

# Decision Hyperplanes and Linear Separability

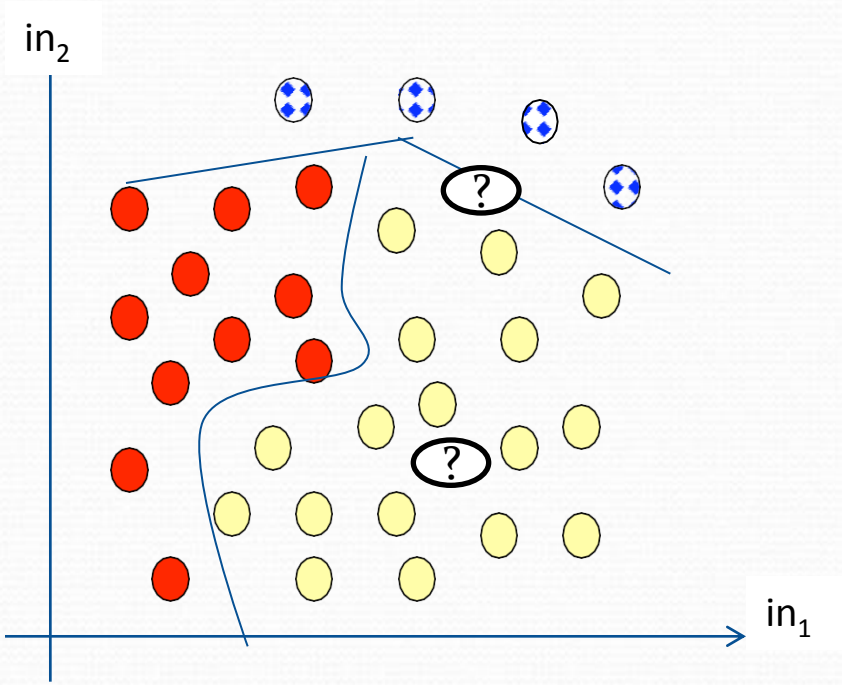
- If user have two inputs, then the weights define a decision boundary that is a one dimensional straight line in the two dimensional ***input space*** of possible input values.
- If we have *n* inputs, the weights define a decision boundary that is an *n*–1 dimensional ***hyperplane*** in the *n* dimensional input space:

$$w_1 in_1 + w_2 in_2 + w_n in_n - \theta = 0$$

- This hyperplane is linear (i.e., straight or flat or non-curved) and can only divide the space into two regions.
- We still need more complex transfer functions, or more complex networks, to deal with XOR type problems.
- Problems with input patterns that can be classified using a single hyperplane are said to be ***linearly separable***. Problems (like XOR) which cannot be classified in this way are said to be ***non-linearly separable***.

# General Decision Boundaries

- To deal with input patterns that are not binary, and expect our neural networks to form complex decision boundaries, e.g.



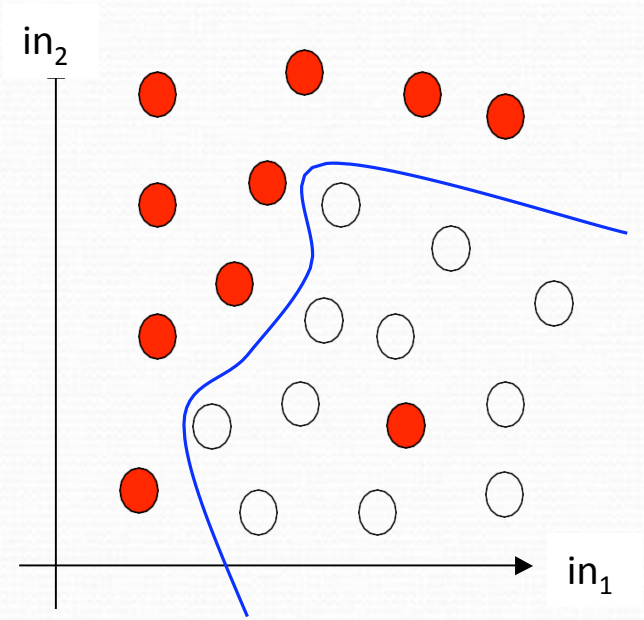

- In figure classes are classified into three categories

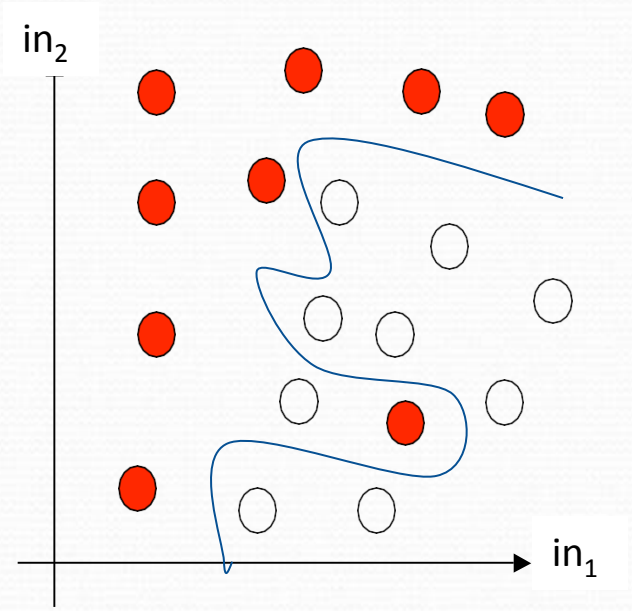
# Learning and Generalization

- A network produce outputs for input patterns that was not originally set up to classify (shown with question marks), though those classifications may be incorrect.
- There are two important aspects of the network's operation :
  - **Learning**: The network must learn decision boundaries from a set of ***training patterns*** so that these training patterns are classified correctly.
  - **Generalization**: After training, the network must also be able to generalize, i.e. correctly classify ***test patterns*** it has never seen before.
- Sometimes, the training data may contain errors (e.g., noise in the experimental determination of the input values, or incorrect classifications). In this case, learning the training data perfectly may make the generalization worse.

# Generalization in Classification

- The objective of the neural network is to learn a classification decision boundary:

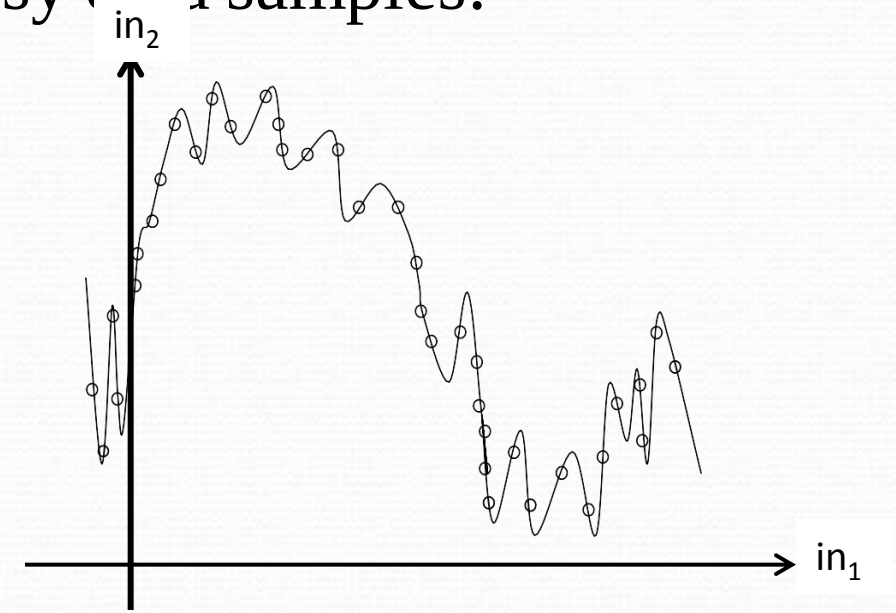



The objective is to get the network to generalize to classify new inputs appropriately. If the training data contains noise, we don't necessarily want the training data to be classified totally accurately, as that is likely to reduce the generalization ability.

# Generalization in Function Approximation

- We wish to recover a function for which we only have noisy data samples:




- The neural network output give better representation of the under-lying function if its output curve does not pass through all the data points.
- A larger error on the training data is likely to lead to better generalization.

# Review Questions

- What is the difference between biological neurons and artificial neurons?
- What is perceptron? What are the different types of perceptron?
- What is decision boundary? Explain.
- What are the two important aspects of network operation?
- What is classification decision boundary?
- What is general decision boundary? Explain.

# References


- List of Books
  - Understanding Machine Learning: From Theory to Algorithms.
  - Introductory Machine Learning notes
  - Foundations of Machine Learning

- List of websites referred:
  - https://www.simplilearn.com/what-is-perceptron-tutorial.
  - https://www.cs.bham.ac.uk/~pxt/NC/l4_JB.pdf
  - https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15381-f01/www/handouts/110601.pdf
  - https://www.simplilearn.com/what-is-perceptron-tutorial


Thank you!