



# **SNS COLLEGE OF TECHNOLOGY**



**AN AUTONOMOUS INSTITUTION**

**Approved by AICTE New Delhi & Affiliated to Anna University Chennai  
Accredited by NBA & Accredited by NAAC with “A+” Grade, Recognized by UGC  
COIMBATORE**

## **DEPARTMENT OF CIVIL ENGINEERING**

### **MACHINE LEARNING FOR CIVIL ENGINEERS**

#### **II YEAR / IV SEMESTER**

##### **Unit 2 : Supervised Learning**

##### **Topic 4: Neural Networks – Back Propagation**



# Introduction

- Artificial Neural Networks are crude attempts to model the highly massive parallel and distributed processing we believe takes place in the brain.
- Back propagation, an abbreviation for "backward propagation of errors", is a common method of training artificial neural networks used in conjunction with an optimization method such as gradient descent.
- The method calculates the gradient of a loss function with respect to all the weights in the network.
- The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function



# Why do we need Multi Layer Neural Networks ?



- Some input and output patterns can be easily learned by single-layer neural networks (i.e. perceptron). However, these single-layer perceptron cannot learn some relatively simple patterns, such as those that are not linearly separable.
- Single layer neural network can't learn any abstract features of the input since it is limited to having only one layer. A multi-layered network overcomes this limitation as it can create internal representations and learn different features in each layer.
- Each higher layer learns more and more abstract features that can be used to classify the image. Each layer finds patterns in the layer below it and it is this ability to create internal representations that are independent of outside input that gives multi-layered networks their power.



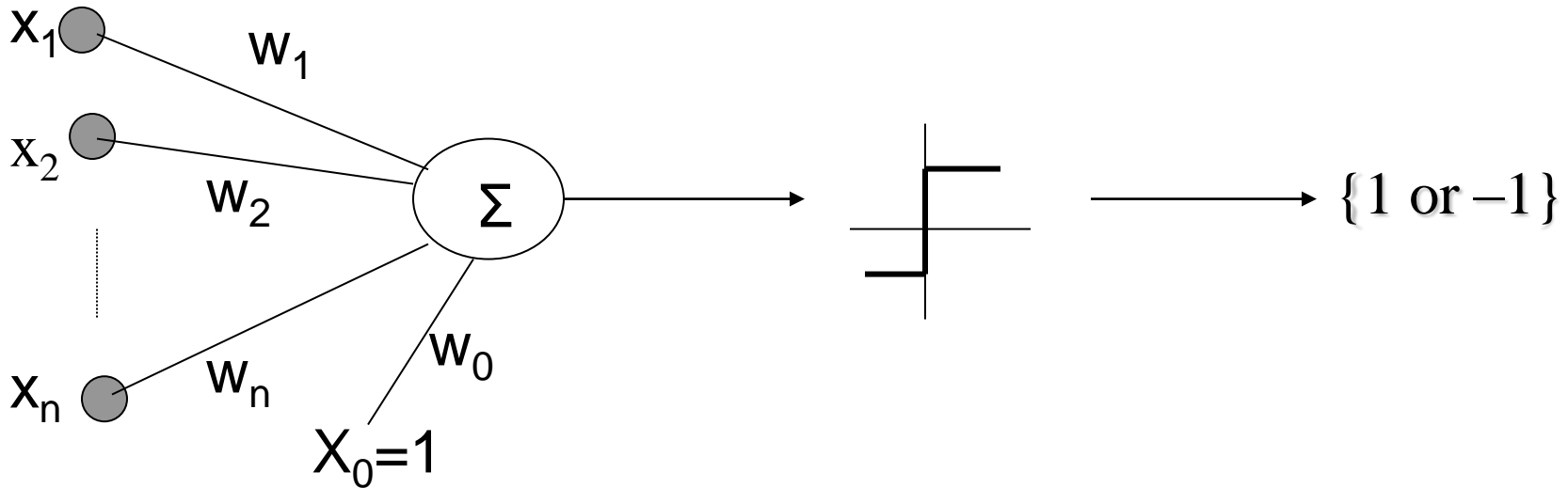
# Historical Background

- Early attempts to implement artificial neural networks: McCulloch (Neuroscientist) and Pitts (Logician) (1943)
- Based on simple neurons (MCP neurons)
- Based on logical functions
- Donald Hebb (1949) gave the hypothesis in his thesis “The Organization of Behavior”:
- “Neural pathways are strengthened every time they are used.”
- Frank Rosenblatt (1958) created the perceptron, an algorithm for pattern recognition based on a two-layer computer learning network using simple addition and subtraction .



# Perceptron

- It is a step function based on a linear combination of real-valued inputs. If the combination is above a threshold it outputs a 1, otherwise it outputs a  $-1$
- A perceptron can only learn examples that are linearly separable





# Delta Rule

The delta rule is used for calculating the gradient that is used for updating the weights.

We will try to minimize the following error:

- $E = \frac{1}{2} \sum_i (t_i - o_i)^2$

For a new training example  $\mathbf{X} = (x_1, x_2, \dots, x_n)$ , update each weight according to this rule:

- $w_i = w_i + \Delta w_i$   
where,  $\Delta w_i = -\eta * E'(w)/w_i$

The derivative gives,

- $E'(w)/w_i = \sum_i (t_i - o_i) * (-x_i)$

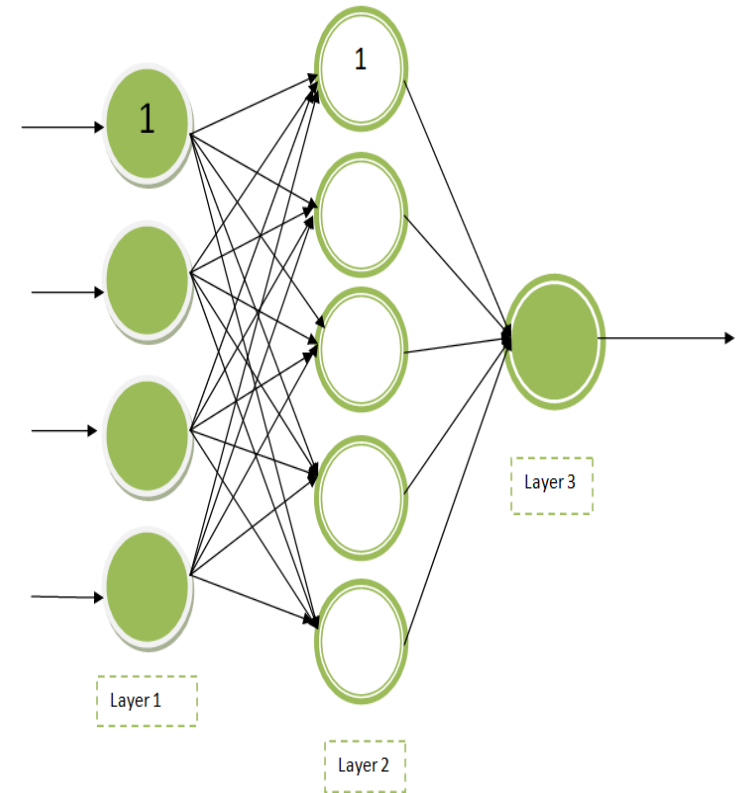
So that gives us the following equation:

- $\Delta w_i = \eta \sum_i (t_i - o_i) x_i$



# Multi Layer Neural Networks

In contrast to perceptrons, multilayer neural networks can not only learn multiple decision boundaries, but the boundaries may be nonlinear.





# Learning non-linear boundaries



To make nonlinear partitions on the space we need to define each unit as a nonlinear function (unlike the perceptron). One solution is to use the sigmoid (logistic) function. So,

$$O(x_1, x_2, \dots, x_n) = \sigma ( WX )$$

$$\text{where: } \sigma ( WX ) = 1 / 1 + e^{-WX}$$

We use the sigmoid function because of the following property,

- $d \sigma(y) / dy = \sigma(y) (1 - \sigma(y))$





# Back propagation Algorithm

**The back propagation learning algorithm can be divided into two phases:**

## **Phase 1: Propagation**

- Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.
- Backward propagation of the propagation's output activations through the neural network using the training pattern target in order to generate the deltas (the difference between the input and output values) of all output and hidden neurons.

## **Phase 2: Weight update**

- Multiply its output delta and input activation to get the gradient of the weight.
- Subtract a ratio (percentage) of the gradient from the weight.



# Hidden Units

- The number of hidden units is related to the complexity of the decision boundary.
- If examples are easy to discriminate few nodes would be enough. Conversely complex problems require many internal nodes.
- A rule of thumb is to choose roughly  $m / 10$  weights, where  $m$  is the number of training examples.



# Learning Rates

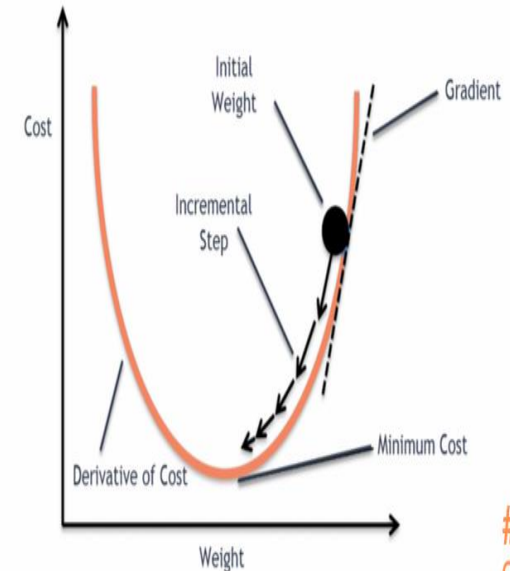
- Different learning rates affect the performance of a neural network significantly.
- Optimal Learning Rate:
- Leads to the error minimum in one learning step.



# What Is Gradient Descent Algorithm??



- Back propagation calculates the gradient of the error of the network regarding the network's modifiable weights.
- This gradient is almost always used in a simple stochastic gradient descent algorithm to find weights that minimize the error.



#MLmuse  
CLAIRVOYANT



# Limitations of the back propagation algorithm



**It is not guaranteed to find global minimum of the error function. It may get trapped in a local minima,**

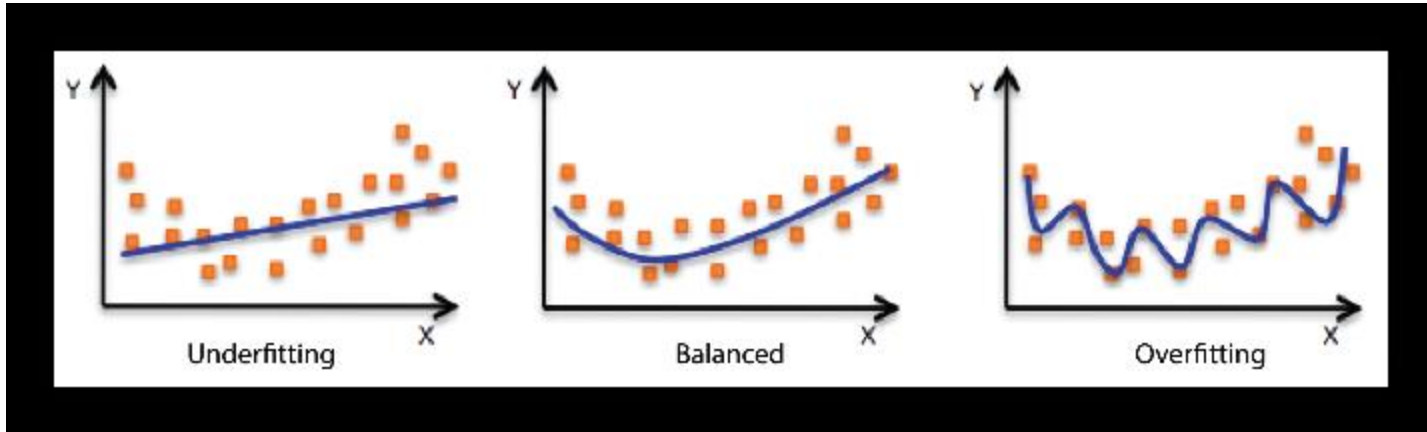
- Improvements,
  - Add momentum.
  - Use stochastic gradient descent.
  - Use different networks with different initial values for the weights.

**Back propagation learning does not require normalization of input vectors; however, normalization could improve performance.**

- Standardize all features previous to training.



# Generalization and Overfitting



## Solutions,

- Use a validation set and stop until the error is small in this set.
- Use 10 fold cross validation



*Thank You!!*