UNIT I INTRODUCTION TO DEEP LEARNING

Introduction to machine learning - Linear models (SVMs and Perceptron's, logistic regression)- Introduction to Neural Nets: What are a shallow network computes- Training a network: loss functions, back propagation and stochastic gradient descent- Neural networks as universal function approximates.

1 Definition of Machine Learning [ML]:

Well posed learning problem: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."(Tom Michel)

Machine Learning (ML) is an algorithm that works on consequently through experience and by the utilization of information. It is viewed as a piece of AI. ML calculations assemble a model dependent on example information (Data), known as "training Data or information", to settle on forecasts or choices without being unequivocally customized to do as such. AI calculations are utilized in a wide assortment of utilizations, for example, in medication, email sifting, discourse acknowledgment, and Computer vision, where it is troublesome or impractical to foster customary models to play out the needed tasks.

ML includes PCs finding how they can perform tasks without being expressly modified to do as such. It includes Systems that can perform tasks without being expressly modified to do as such. It includes models gaining data so they can do certain specific applications. For basic undertakings appointed to Models, it is feasible to models advising the machine how to execute all means needed to tackle the current issue; on the systems part, no learning is required. For further developed undertakings, it tends to be trying for a human to physically make the required calculations. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step.

The term *machine learning* was coined in 1959 by Arthur Samuel, an American IBMer and pioneer in the field of computer gaming and artificial intelligence.

1.1 Fundamentals of ANN

Neural computing is an information processing paradigm, inspired by biological system, composed of a large number of highly interconnected processing elements(neurons) working in unison to solve specific problems.

Artificial neural networks (ANNs), like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

1.2 The Biological Neuron

The human brain consists of a large number, more than a billion of neural cells that process information. Each cell works like a simple processor. The massive interaction between all cells and

their parallel processing only makes the brain's abilities possible. Figure 1 represents a human biological nervous unit. Various parts of biological neural network(BNN) is marked in Figure 1.

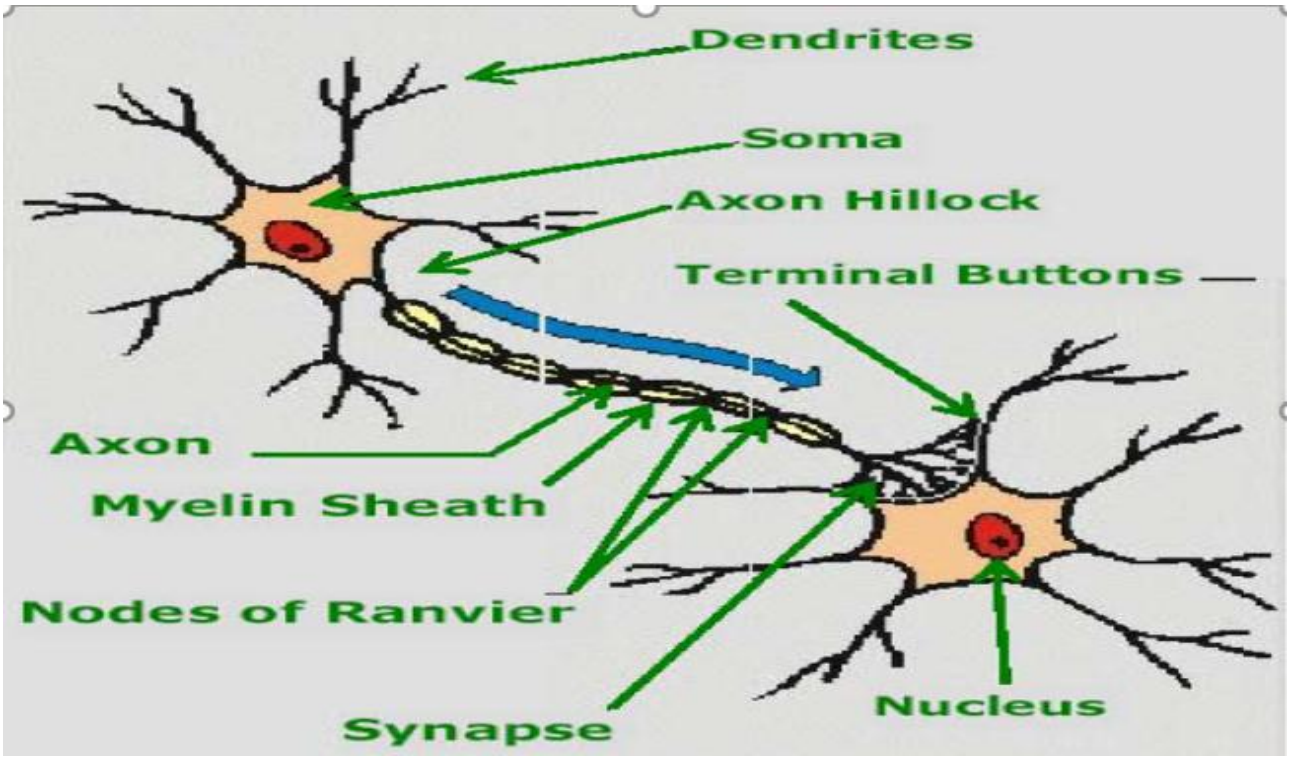

Figure 1: Biological Neural Network

Dendrites are branching fibres that extend from the cell body or soma.

Soma or cell body of a neuron contains the nucleus and other structures, support chemical processing and production of neurotransmitters.

Axon is a singular fiber carries information away from the soma to the synaptic sites of other neurons (dendrites ans somas), muscels, or glands.

Axon hillock is the site of summation for incoming information. At any moment, the collective influence of all neurons that conduct impulses to a given neuron will determine whether or n ot an action potential will be initiated at the axon hillock and propagated along the axon.

Myelin sheath consists of fat-containing cells that insulate the axon from electrical activity. This insulation acts to increase the rate of transmission of signals. A gap exists between each myelin sheath cell along the axon. Since fat inhibits the propagation of electricity, the signals jump from one gap to the next.

Nodes of Ranvier are the gaps (about 1 μm) between myelin sheath cells. Since fat serves as a good insulator, the myelin sheaths speed the rate of transmission of an electrical impulse along the axon.

Synapse is the point of connection between two neurons or a neuron and a muscle or a gland. Electrochemical communication between neurons take place at these junctions.

Terminal buttons of a neuron are the small knobs at the end of an axon that release chemicals called neurotransmitters.

Information flow in a neural cell

The input/output and the propagation of information are shown below.

1.3. Artificial neuron model

An artificial neuron is a mathematical function conceived as a simple model of a real (biological) neuron.

- The McCulloch-Pitts Neuron
  This is a simplified model of real neurons, known as a Threshold Logic Unit.
- A set of input connections brings in activations from other neuron.
- A processing unit sums the inputs, and then applies a non-linear activation function (i.e. squashing/transfer/threshold function).
- An output line transmits the result to other neurons.

1.3.1 Basic Elements of ANN:

Neuron consists of three basic components –weights, thresholds and a single activation function. An Artificial neural network(ANN) model based on the biological neural sytems is shown in figure 2.

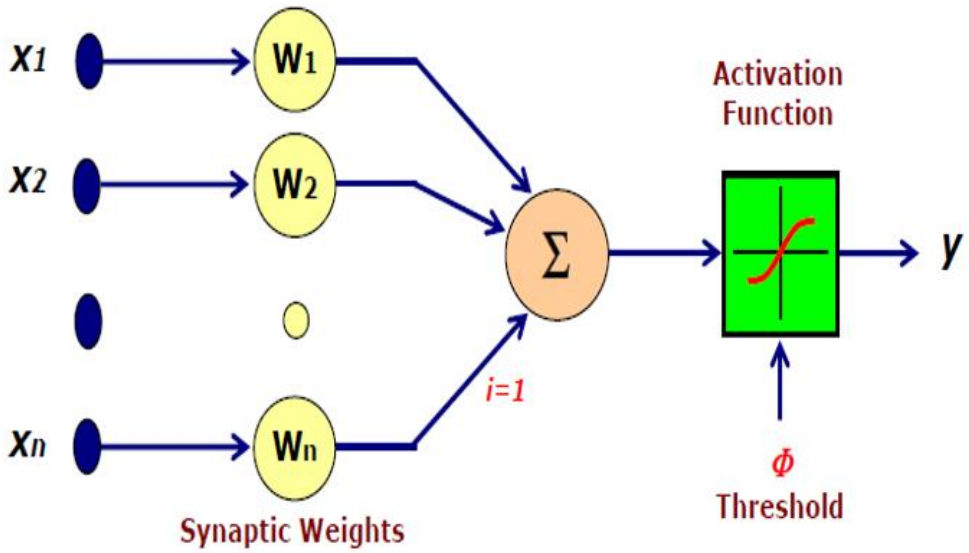

Figure 2: Basic Elements of Artificial Neural Network

1.4 Different Learning Rules

A brief classification of Different Learning algorithms is depicted in figure 3.

- **Training: It is the process in which the network is taught to change its weight and bias.**
- **Learning: It is the internal process of training where the artificial neural system learns to update/adapt the weights and biases.**

Different Training /Learning procedure available in ANN are

- **Supervised learning**
- **Unsupervised learning**
- **Reinforced learning**
- **Hebbian learning**
- **Gradient descent learning**
- **Competitive learning**
- **Stochastic learning**

1.4.1. Requirements of Learning Laws:

- **Learning Law should lead to convergence of weights**
- **Learning or training time should be less for capturing the information from the training pairs**
- **Learning should use the local information**
- **Learning process should able to capture the complex non linear mapping available between the input & output pairs**
- **Learning should able to capture as many as patterns as possible**
- **Storage of pattern information's gathered at the time of learning should be high for the given network**

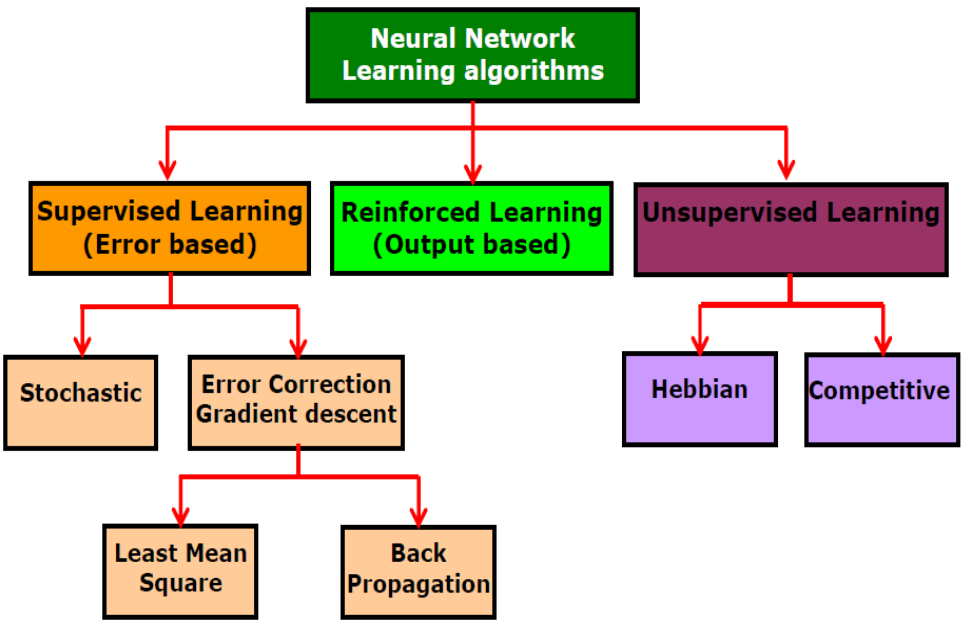

Figure 3: Different Training methods of Artificial Neural Network

1.4.1.1.Supervised learning :

Every input pattern that is used to train the network is associated with an output pattern which is the target or the desired pattern.

A teacher is assumed to be present during the training process, when a comparison is made between the network's computed output and the correct expected output, to determine the error.The error can then be used to change network parameters, which result in an improvement in performance.

1.4.1.2 Unsupervised learning:

In this learning method the target output is not presented to the network.It is as if there is no teacher to present the desired patterns and hence the system learns of its own by discovering and adapting to structural features in the input patterns.

1.4.1.3 Reinforced learning:

In this method, a teacher though available, doesnot present the expected answer but only indicates if the computed output correct or incorrect.The information provided helps the network in the learning process**.**

1.4.1.4 Hebbian learning:

This rule was proposed by Hebb and is based on correlative weight adjustment.This is the oldest learning mechanism inspired by biology.In this, the input-output pattern pairs (x_i, y_i) are associated by the weight matrix W, known as the correlation matrix.

It is computed as

$$W = \sum_{i=1}^{n} x_i y_i^T \quad \text{---------- eq(1)}$$

Here y_i^T is the transposeof the associated output vector y_i.Numerous variants of the rule have been proposed.

1.4.1.5 Gradient descent learning:

This is based on the minimization of error E defined in terms of weights and activation function of the network.Also it is required that the activation function employed by the network is differentiable, as the weight update is dependent on the gradient of the error E.

Thus if Δw_{ij} is the weight update of the link connecting the i^{th} and j^{th} neuron of the two neighbouring layers, then Δw_{ij} is defined as,

$$\Delta w_{ij} = \eta \frac{\partial E}{\partial w_{ij}} \quad \text{---------- eq(2)}$$

Where, η is the learning rate parameter and $\frac{\partial E}{\partial w_{ij}}$ is the error gradient with reference to the weight w_{ij}.

1.5 Perceptron Model

1.5.1 Simple Perceptron for Pattern Classification

Perceptron network is capable of performing pattern classification into two or more categories. The perceptron is trained using the perceptron learning rule. We will first consider classification into two categories and then the general multiclass classification later. For classification

into only two categories, all we need is a single output neuron. Here we will use bipolar neurons. The simplest architecture that could do the job consists of a layer of N input neurons, an output layer with a single output neuron, and no hidden layers. This is the same architecture as we saw before for Hebb learning. However, we will use a different transfer function here for the output neurons as given below in eq (7). Figure 7 represents a single layer perceptron network.

$$y = \begin{cases} 1 & \text{if } y\_in > \theta \\ 0 & \text{if } -\theta \le y\_in \le \theta \\ -1 & \text{if } y\_in < -\theta \end{cases}$$

---------- eq (7)

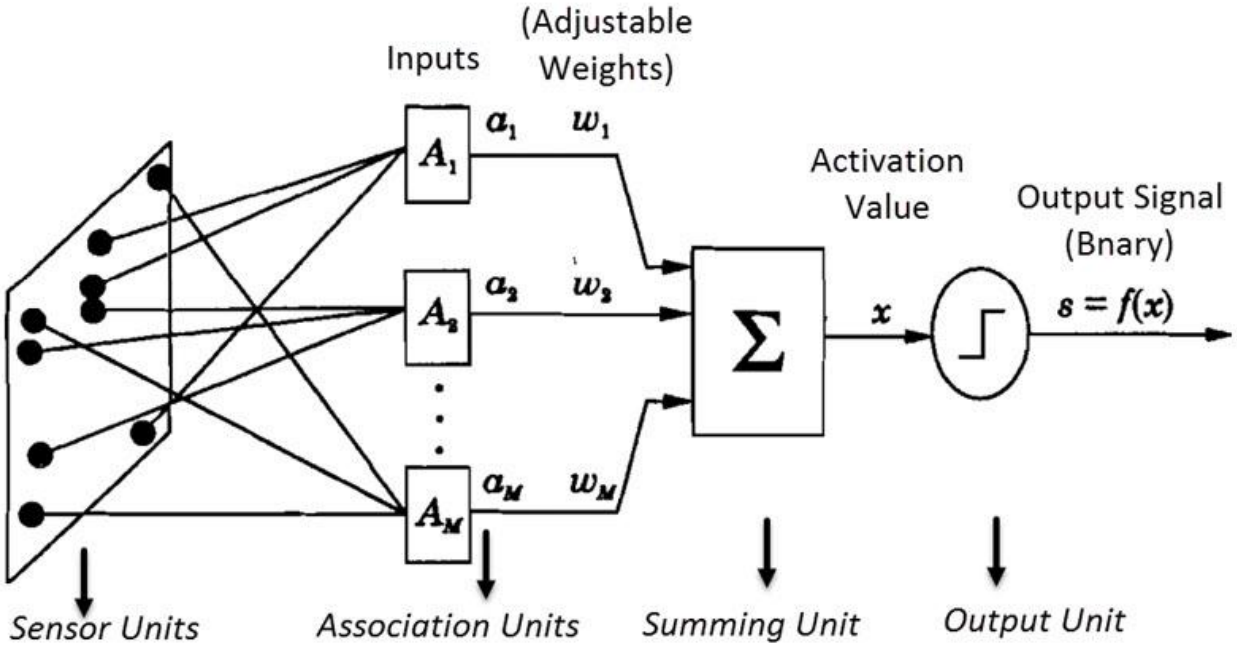

Figure 4: Single Layer Perceptron

Equation 7 gives the bipolar activation function which is the most common function used in the perceptron networks. Figure 7 represents a single layer perceptron network. The inputs arising from the problem space are collected by the sensors and they are fed to the aswociation units.Association units are the units which are responsible to associate the inputs based on their similarities. This unit groups the similar inputs hence the name association unit. A single input from each group is given to the summing unit.Weights are randomnly fixed intially and assigned to this inputs. The net value is calculate by using the expression

$$x = \Sigma w_i a_i - \theta \quad \text{eq(8)}$$

This value is given to the activation function unit to get the final output response.The actual output is compared with the Target or desired .If they are same then we can stop training else the weights haqs to be updated .It means there is error .Error is given as **δ = b-s** , where b is the desired

/ Target output and S is the actual outcome of the machinehere the weights are updated based on the perceptron Learning law as given in equation 9.

Weight change is given as $\Delta w = \eta \delta a_i$. So new weight is given as

$$W_{i\,(new)} = W_{i\,(old)} + \text{Change in weight vector } (\Delta w) \_\_\_\_\_ \text{eq(9)}$$

1.5.2. Perceptron Algorithm

Step 1: Initialize weights and bias.For simplicity, set weights and bias to zero.Set learning rate in the range of zero to one.

- Step 2: While stopping condition is false do steps 2-6
- Step 3: For each training pair s:t do steps 3-5
- Step 4: Set activations of input units $x_i = ai$
- Step 5: Calculate the summing part value Net = Σ aiwi-θ
- Step 6: Compute the response of output unit based on the activation functions
- Step 7: Update weights and bias if an error occurred for this pattern(if *yis not equal to t)*

  $$Weight\ (new) = w_i(old) + atx_i\ ,\ \&\ bias\ (new) = b(old) + at$$

  $$\text{Else } w_i(new) = w_i(old)\ \&\ b(new) = b(old)$$

- Step 8: Test Stopping Condition

1.5.3. Limitations of single layer perceptrons:

- Uses only Binary Activation function
- Can be used only for Linear Networks
- Since uses Supervised Learning ,Optimal Solution is provided
- Training Time is More
- Cannot solve Linear In-separable Problem

1.5.4. Multi-Layer Perceptron Model:

Figure 8 is the general representation of Multi layer Perceptron network.Inbetween the input and output Layer there will be some more layers also known as Hidden layers.

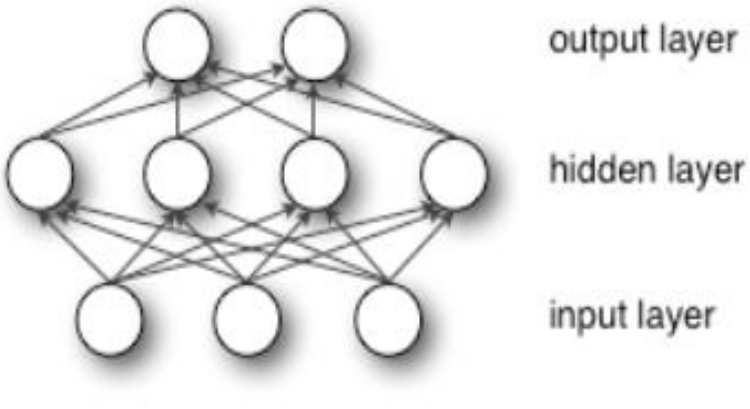

Figure 5: Multi-Layer Perceptron

1.5.5. Multi Layer Perceptron Algorithm

1. Initialize the weights (W_i) & Bias (B_0) to small random values near Zero
2. Set learning rate η or α in the range of "0" to "1"
3. Check for stop condition. If stop condition is false do steps 3 to 7
4. For each Training pairs do step 4 to 7
5. Set activations of Output units: $x_i = s_i$ for i=1 to N
6. Calculate the output Response

   $y_{in} = b_0 + \Sigma x_i w_i$
7. Activation function used is Bipolar sigmoidal or Bipolar Step functions

   For Multi Layer networks, based on the number of layers steps 6 & 7 are repeated
8. If the Targets is (not equal to) = to the actual output (Y), then update weights and bias based on Perceptron Learning Law

   $W_{i\,(new)} = W_{i\,(old)}$ + Change in weight vector

   Change in weight vector $= \eta t_i x_i$

   Where η = Learning Rate

   t_i = Target output of i^{th} unit

   x_i = i^{th} Input vector

   $b_{0(new)} = b_{0\,(old)}$ + Change in Bias

   Change in Bias = ηt_i

   Else $W_{i\,(new)} = W_{i\,(old)}$

   $b_{0(new)} = b_{0\,(old)}$
9. Test for Stop condition

1.6. linearly seperable & Linear in separable tasks:

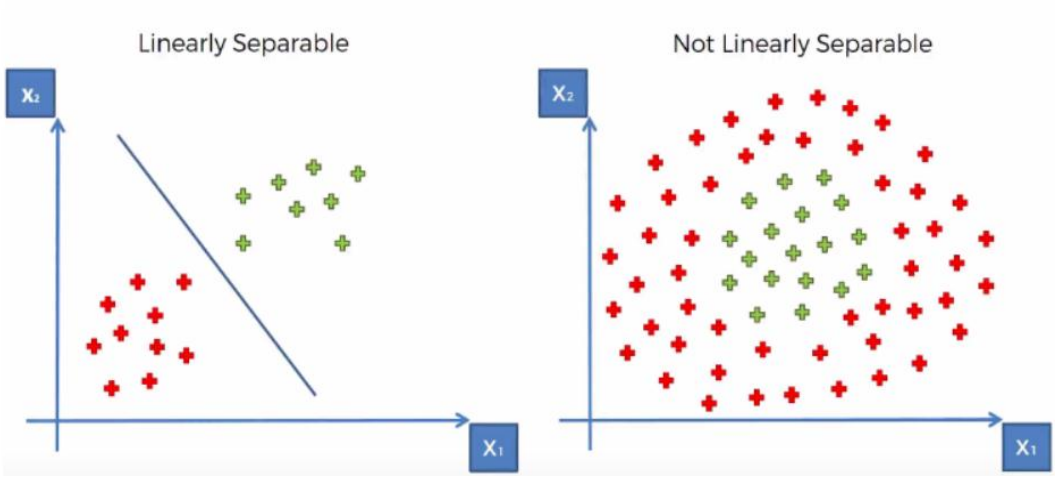

Figure 6: Representation of Linear seperable & Linear-in separable Tasks

Perceptron are successful only on problems with a linearly separable solution sapce.Figure 9 represents both linear separable as well as linear in seperable problem.Perceptron cannot handle, in particular, tasks which are not linearly separable.(Known as linear inseparable problem).Sets of points in two dimensional spaces are linearly separable if the sets can be seperated by a straight line.Generalizing, a set of points in n-dimentional space are that can be seperated by a straight line.is called Linear seperable as represented in figure 9.

Single layer perceptron can be used for linear separation.Example AND gate.But it cant be used for non linear ,inseparable problems.(Example XOR Gate).Consider figure 10.

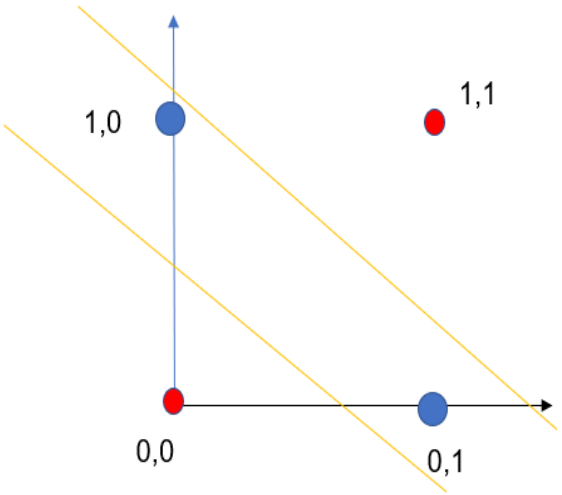

Figure 7: XOR representation (Linear-in separable Task)

Here a single decision line cannot separate the Zeros and Ones Linearly.At least Two lines are required to separate Zeros and Onesas shown in Figure 10. Hence single layer networks can not be used to solve inseparable problems. To over come this problem we go for creation of **convex regions**.

Convex regions can be created by multiple decision lines arising from multi layer networks.Single layer network cannot be used to solve inseparable problem.Hence we go for multilayer network there by creating convex regions which solves the inseparable problem.

1.6.1 Convex Region:

Select any Two points in a region and draw a straight line between these two points. If the points selected and the lines joining them both lie inside the region then that region is **known as convex regions.**

1.6.2. Types of convex regions

(a) Open Convex region (b) Closed Convex region

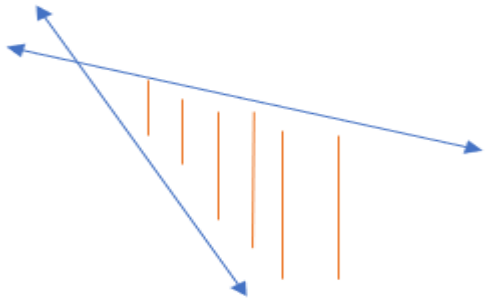

Figure 8: Open convex region

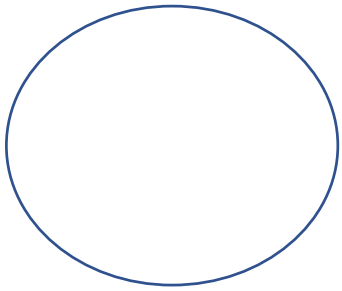

Figure 9 A: Circle - Closed convex region

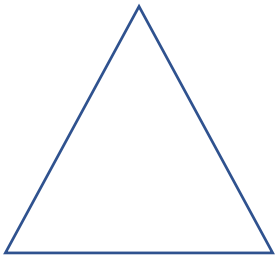

Figure 9 B: Triangle - Closed convex region

1.7. Logistic Regression

Logistic regression is a probabilistic model that organizes the instances in terms of probabilities. Because the classification is probabilistic, a natural method for optimizing the parameters is to ensure that the predicted probability of the observed class for each training occurrence is as large as possible. This goal is achieved by using the notion of maximumlikelihood estimation in order to learn the parameters of the model. The likelihood of the training data is defined as the product of the probabilities of the observed labels of each training instance. Clearly, larger values of this objective function are better. By using the negative logarithm of this value, one obtains a loss function in minimization form. Therefore, the output node uses the negative log-likelihood as a loss function. This loss function replaces the squared error used in the Widrow-Hoff method. The output layer can be formulated with the sigmoid activation function, which is very common in neural network design.

- Logistic regression is another supervised learning algorithm which is used to solve the classification problems. In classification problems, we have dependent variables in a binary or discrete format such as 0 or 1.

- Logistic regression algorithm works with the categorical variable such as 0 or 1, Yes or No, True or False, Spam or not spam, etc.

- It is a predictive analysis algorithm which works on the concept of probability.

- Logistic regression is a type of regression, but it is different from the linear regression algorithm in the term how they are used.

- Logistic regression uses sigmoid function or logistic function which is a complex cost function. This sigmoid function is used to model the data in logistic regression. The function can be represented as:

$$f(x)= \frac{1}{1+e^{-x}}$$

Where f(x)= Output between the 0 and 1 value.
x= input to the function
e= base of natural logarithm.

When we provide the input values (data) to the function, it gives the S-curve as follows: It uses the concept of threshold levels, values above the threshold level are rounded up to 1, and values below the threshold level are rounded up to 0.

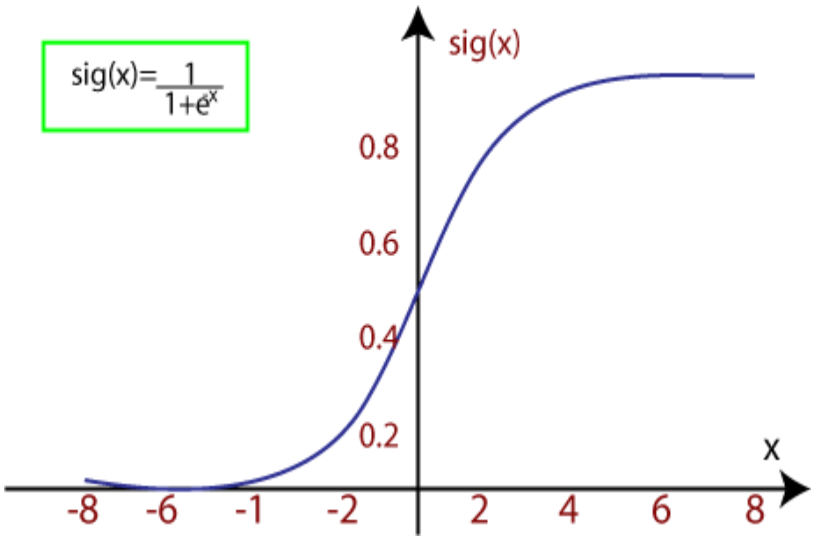

Figure 10: Circle – Logistic Function

1.8. Support Vector Machines

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

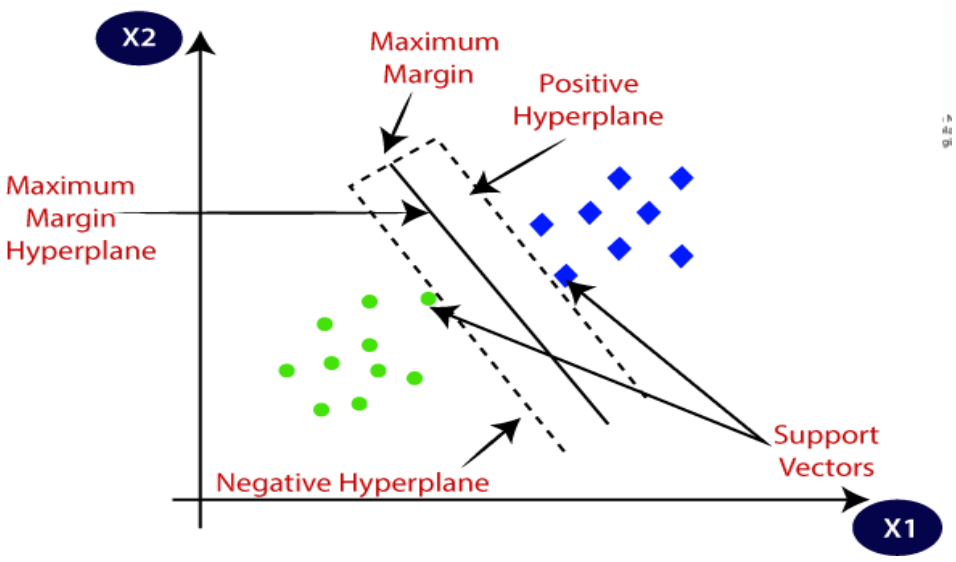

Figure 11: SVM – Classification

1.8.1.SVM can be of two types:

Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

1.8.2. Linear SVM:

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair (x1, x2) of coordinates in either green or blue. Consider the below image figure11. It is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

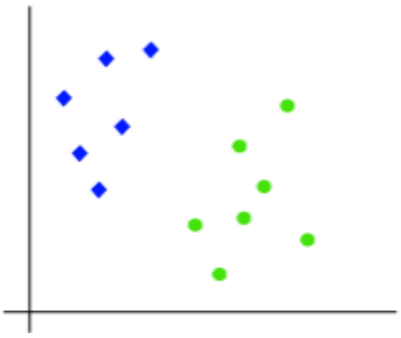

Figure 12A: SVM – Input Space

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a hyperplane. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as margin. And the goal of SVM is to maximize this margin. The hyperplane with maximum margin is called the optimal hyperplane.

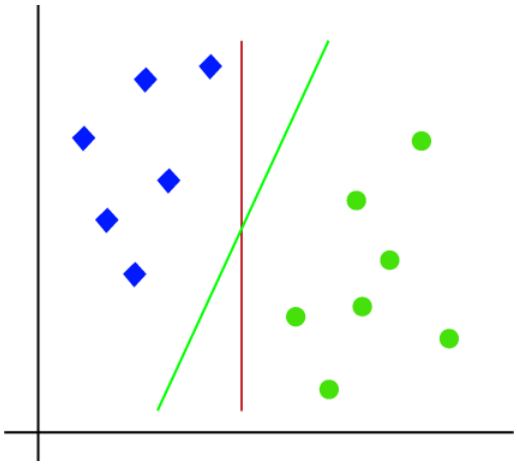

Figure 12B: SVM – Linear Classification

1.9. Gradient Descent:

Gradient Descent is a popular optimization technique in Machine Learning and Deep Learning, and it can be used with most, if not all, of the learning algorithms. A gradient is the slope of a function. It measures the degree of change of a variable in response to the changes of another variable. Mathematically, Gradient Descent is a convex function whose output is the partial derivative of a set of parameters of its inputs. The greater the gradient, the steeper the slope.Starting from an initial value, Gradient Descent is run iteratively to find the optimal values of the parameters to find the minimum possible value of the given cost function.

1.9.1. Types of Gradient Descent:

Typically, there are three types of Gradient Descent:

1. Batch Gradient Descent
2. Stochastic Gradient Descent
3. Mini-batch Gradient Descent

1.9.2. Stochastic Gradient Descent (SGD):

The word ‘stochastic‘ means a system or a process that is linked with a random probability. Hence, in Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration. In Gradient Descent, there is a term called “batch” which denotes the total number of samples from a dataset that is used for calculating the gradient for each iteration. In typical Gradient Descent optimization, like Batch Gradient Descent, the batch is taken to be the whole dataset. Although, using the whole dataset is really useful for getting to the minima in a less noisy and less random manner, but the problem arises when our datasets gets big. Suppose, you have a million samples in your dataset, so if you use a typical Gradient Descent optimization technique, you will have to use all of the one million samples for completing one iteration while performing the Gradient Descent, and it has to be done for every iteration until the minima is reached. Hence, it becomes computationally very expensive to perform.

Reference Books:

1. B. Yegnanarayana, “Artificial Neural Networks” Prentice Hall Publications.
2. Simon Haykin, “Artificial Neural Networks”, Second Edition, Pearson Education.
3. Laurene Fausett, “Fundamentals of Neural Networks, Architectures, Algorithms and Applications”, Prentice Hall publications.
4. Cosma Rohilla Shalizi, Advanced Data Analysis from an Elementary Point of View, 2015.
5. 2. Deng & Yu, Deep Learning: Methods and Applications, Now Publishers, 2013.
6. 3. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.
7. 4. Michael Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.

Note: For further reference, kindly refer the class notes, PPTs, Video lectures available in the Learning Management System (Moodle)

********** ALL THE BEST **********