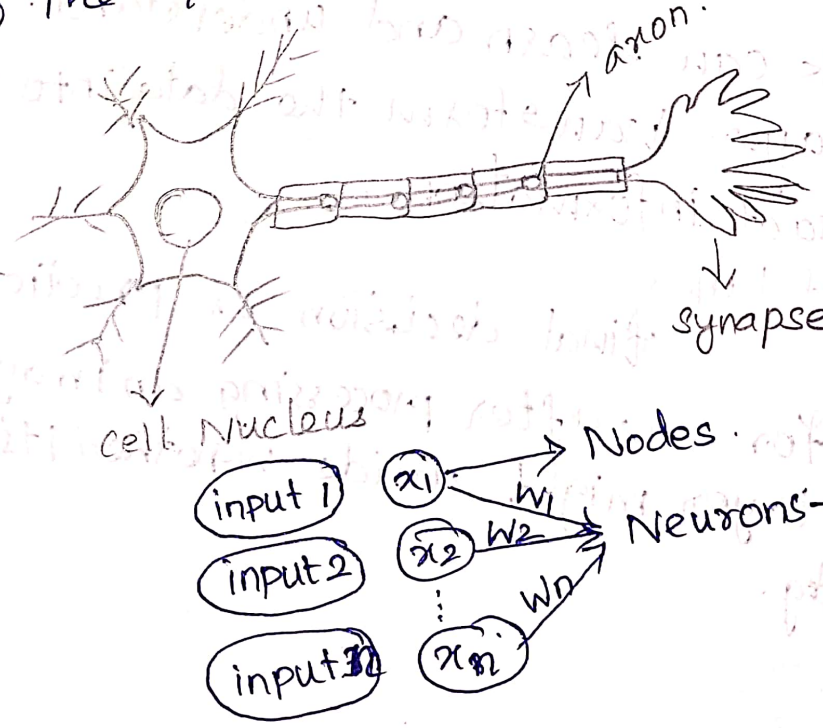
UNIT - 5

NEURAL NETWORKS.

Perceptron - Multilayer perceptron, activation function, n/w training - gradient descent optimization - stochastic gradient descent, error back propagation, from shallow n/w to deep n/w - unit saturation (vanishing gradient problem) - ReLU, hyperparameter tuning, batch normalization, regularization, dropout.

Neural Networks :-

The term "Artificial Neural Network" is derived from Biological Neural N/w that develop the structure of human brain. Similar to the human brain that has neurons interconnected to one another, artificial Neural n/w also have neurons that are inter connected to one another in various layer of the n/w. These neuron are known as nodes.



BNN	ANN.
Dendrities	Inputs.
cell nucleus.	Nodes.
Synapse	weights.
Axon	output.

Artificial Neural Network:-

ANN are the computing system that is designed to simulate the way the human brain analyzes and process the information.

Key Components of an ANN:-

i) input layer:-

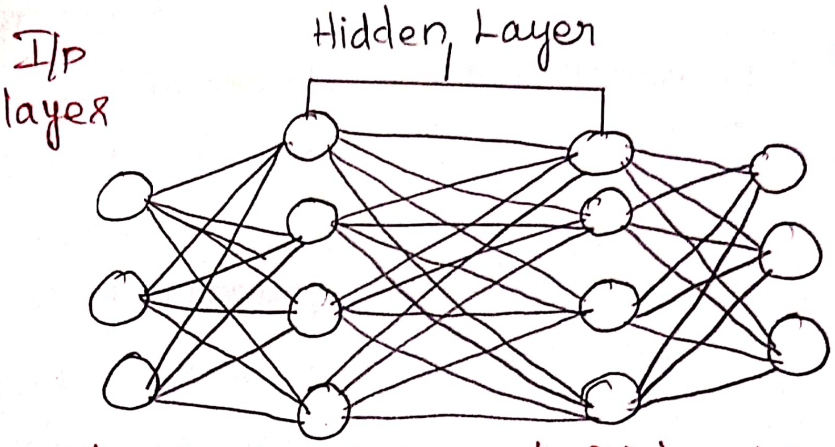
This is where the n/w receive information ex: In an image recognition task, the i/p could be an image.

ii) Hidden Layer:- These layer process the data received from the input layer. The more hidden layers.

These are more complex patterns the network can learn and understand. Each hidden layer transform the data into more abstract information.

iii) Output layers:-

final decision or prediction is made. for ex: after processing an image, the output layer might decide whether its Cat or dg.

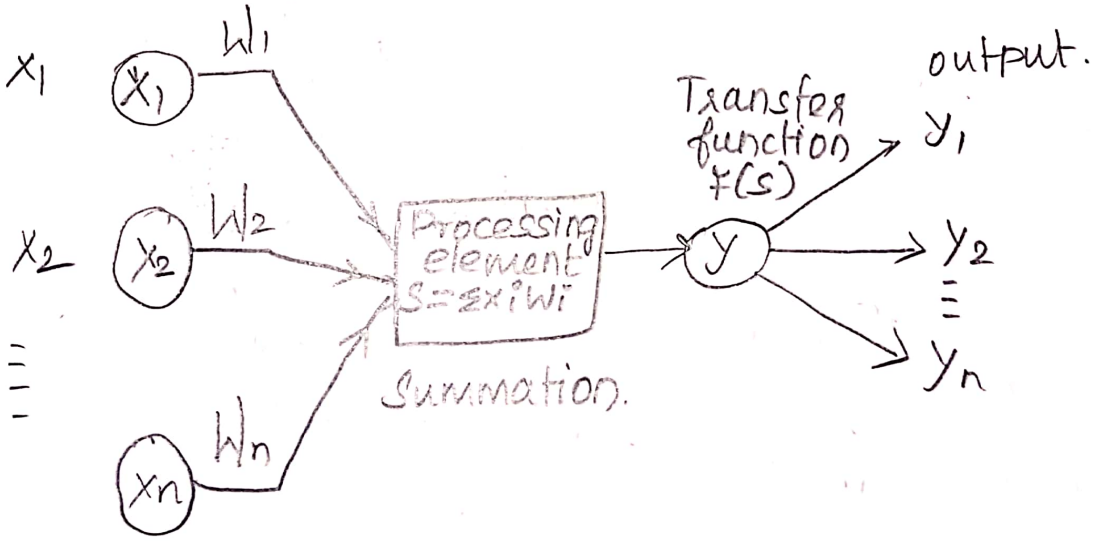


Working of Artificial Neural Network:-

Instead of directly getting into the working of Artificial Neural Networks, lets breakdown and try to understand Neural Networks basic unit, which is called perceptron.

Perceptron can be defined as a neural network with a single layer that classifies the linear data. It further constitutes of 4 major components.

i) Inputs (ii) Weights and Bias.
iii) Summation function (iv) Activation or transformation function.



Perceptron:-

Perceptron is Machine Learning algorithm for Supervised learning of various binary classification tasks.

Perceptron is also understood as a artificial neuron or neural n/w unit that helps to detect certain i/p data computation in business intelligence.

Perceptron model is also treated as one of the best and simplest types of ANN. It is a supervised learning algorithm of binary classifiers. Hence we can consider it as a single layer neural n/w with four main parameters.

* i/p values, * weight / Bias * Netsum
* An Activation function.

Binary classifier:-

In ML Binary classifier are defined as the function that helps in deciding whether i/p data can be represented as vector of numbers and belongs to some specific class.

Binary classifier can be considered as linear classifier. In simple word, we can understand it as a classification algorithm that can Predict linear predictor function in terms of weight and feature vectors.

Perceptron function :-

Perceptron function $f(x)$ can be achieved as output by multiplying the input x with the learned weight co-efficient 'w'

$f(x) = 1$; if $w \cdot x + b > 0$.

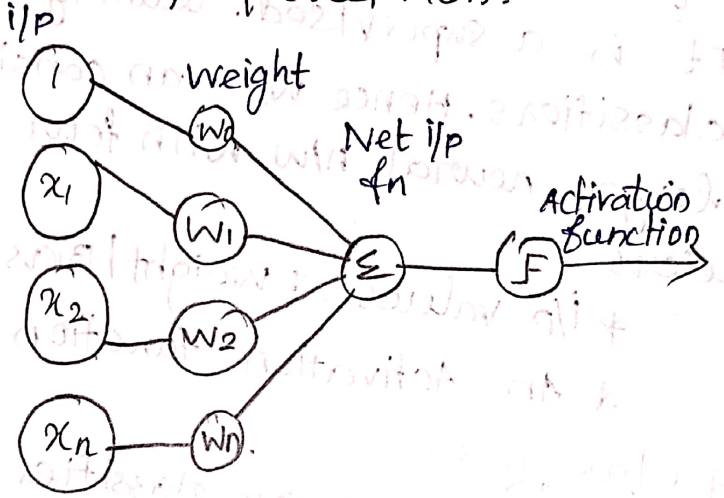
otherwise $f(x) = 0$.

w represent real valued weights vector.

b represent the bias.

x represent a vector of i/p x values.

Components of perceptron :-



Input nodes or Input layers :-

This is primary component of perceptron which accepts the initial data into the s/m for further processing. Each i/p node contains a real numerical value.

Weight and Bias :-

weight parameter represent the strength of the connection b/w units. This is another most important parameter of Perceptron. Components. weight is directly proportional to the strength of the associated input neuron is

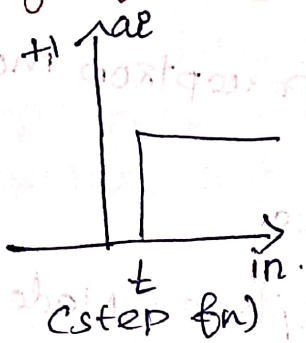
deciding the output. Further, Bias can be considered as the line of intercept in a linear equation.

Activation function:-

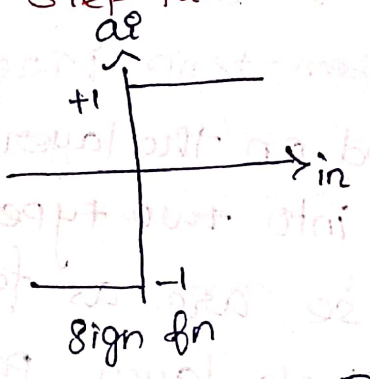
These are final & important components that help to determine whether the neuron will fire or not. Activation function can be considered primarily as a step function.

Types of Activation function:

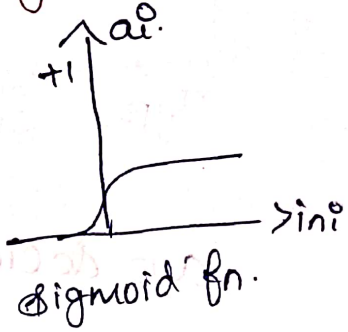
Sign function



Step function

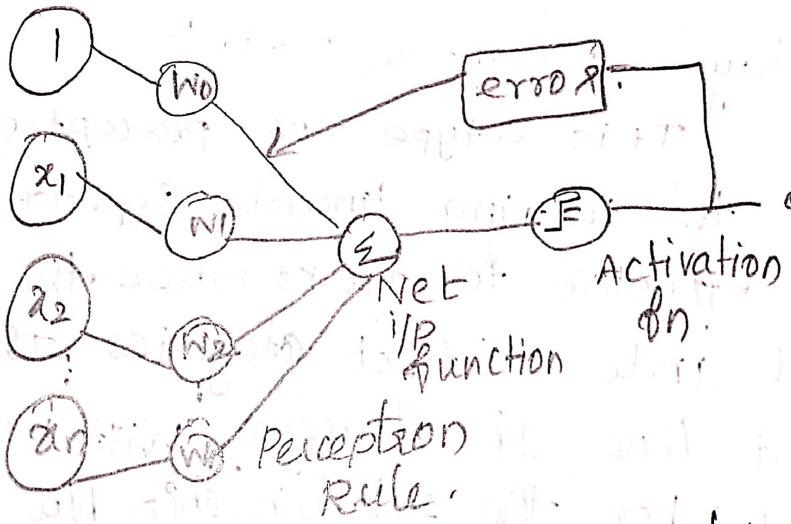


Sigmoid function.



Relu - overcom vanishing gradient problem to avoid.

Working of Perceptron:-



Perceptron model works in two important.

Steps:-

Step1:- In 1st step multiply all i/p values with corresponding weight values and then add them to determine the weighted sum. Calculates the weighted sum as follows.

$\sum w_i * x_i = x_1 * w_1 + x_2 * w_2 + \dots w_n * x_n$

Add a special term called bias 'b' to this weighted sum to improve the models performance.

$\sum w_p * x_i + b$

Step 2:-

In the second step, an activation functions is applied with above-mentioned weighted sum, which gives us output either in binary form or a continuous value as follows.

$$y = f\left(\sum w_i * x_i + b\right)$$

Types of perceptron Models:-

Based on the layer, perceptron models are decided into two types.

These are as follows.

i) Single layer perception Model

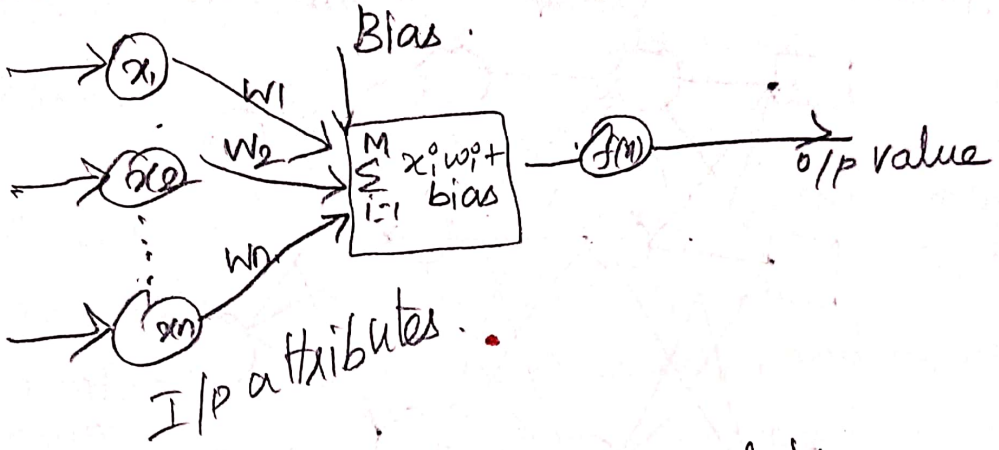
ii) Multi layer Perceptron Model.

Single layer perceptron Model:-

It is a type of perceptron is limited to learning linearly seperable patterns. It is effective for tasks where the data can be divided into distinct categories through a straight line. It struggles with more complex problems where the relationship b/w i/p & o/p is non linear.

A single layered perceptron model consist feed forward network and also include a threshold transfer function inside the model.

The main objective of the single layer Perceptron model is to analyze the linearly seperable object with binary outcomes.



Multi layered Perceptron Model:-

Like a single layer perceptron model a multi layer perceptron model also has the same model structure but has a greater number of hidden layer.

Multilayer perceptron define the most sample architecture of artificial neural network.

The multi layer perceptron model is also known as the back propagation algorithm, which executes in two stages.

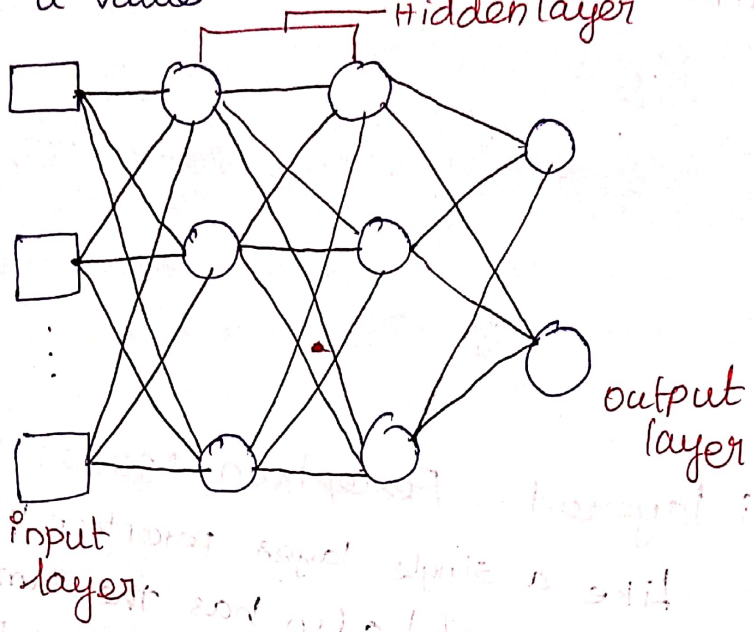
Forward stage:-

Activation function start from the input layers in the forward stage and terminate on the output layer.

Backward stage:-

In the backward stage, weight and bias values are modified as per the models requirement. In this stage the error b/w actual output and demanded originated backward on the output layer and ended on the input layer.

The i/ps are pushed through the ML
the dot product of the input with the we
exist b/w input and the hidden layer. This d
yields a value at the hidden layer.



The hidden layer neither directly receive inputs nor send output to the external environment

The final layer is the OP layer which outputs a single value or a vector of values.

MLP model has considered as multiple AMV having various layers in which activation function does not remain linear, similar to the SLP model.

Instead of linear, activation function can be executed as sigmoid, TanH, Relu etc... for development.

A Multilayer perceptron model has greater processing power and can process linear and non linear patterns. it can also implement logic gates such as: AND, OR, XOR, NAND, NOT XNOR, NOR.

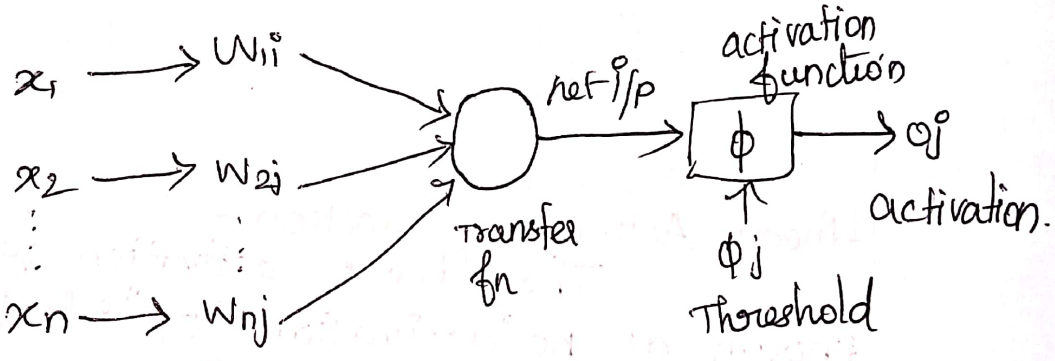
Adv
- Versatility
- Non-linearity
- Parallel Computation

DisAdv
- computationally expensive
- overfitting
- Sensitivity to data scaling.

Activation function:-

The NN Activation fn's are the most significant component of deep learning, they are fundamentally used for determining the output of deep learning models. its accuracy, and Performance efficiency of the training model that can design or divide a huge scale Neural Network.

Activation function also helps to normalize the o/p of any i/p in range b/w 1 to -1 Activation function must be efficient and it should reduce the computation time because the neural network sometimes trained on data points.



The neuron is basically a weighted average of i/p, then this sum is passed through an activation function to get an output.

$$y = \Sigma (\text{weights} * \text{i/p} + \text{bias})$$

These y can be anything for a narrow b/w range - infinity to + infinity, so we have to bound our o/p to get the desired prediction or

generalized results.

y = Activation function $\leq$ (weights * input + bias)

So we pass that neuron to activation function to bound output values.

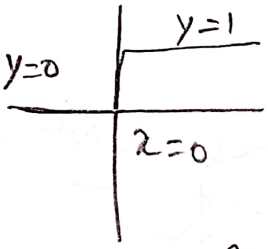
Types of Activation function :-

i) Binary step function
ii) Linear activation function
iii) Non-linear Activation function

Binary step function :-

A binary step function is generally used in the perceptron linear classifier. It threshold the input values to 1 and 0. if they are greater or less than zero respectively.

The step function is mainly used in binary classification and it can't classify the multiclass problems.



$f(x) = 1$; if $x > 0$

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

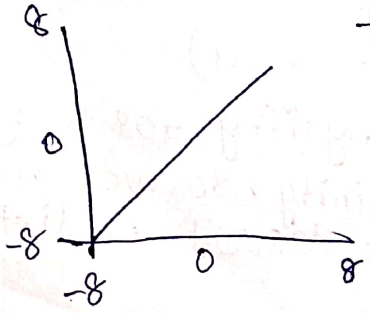
Linear Activation function :-

The linear activation function, also known as no "activation" or "identity function" (multiplied $x_1.0$) is where the activation is proportional to the input.

The equation for linear activation function

$f(x) = a.x.$

When $a=1$ then $f(x) = x$ and this is a special case known as identity.



Linear Activation Function:-

Modern neural networks models use non-linear activation functions. They allow the model to create complex mappings b/w the n/w i/p & o/p such as images video, audio and data set that are non-linear or have high dimensionality.

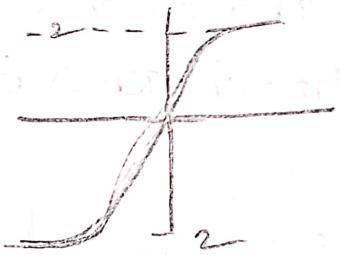
Three types of non linear Activation function

1) Sigmoid Activation Function
2) ReLu (Rectified Linear units)
3) Complex Non-linear Activation function.

Sigmoid Activation function:-

It is characterized by s shape. This function takes by any real value as i/p & o/p values in the range of 0 to 1 hence useful for binary classification.

The function exhibits a steep gradient when x values are b/w -2 and 2.



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

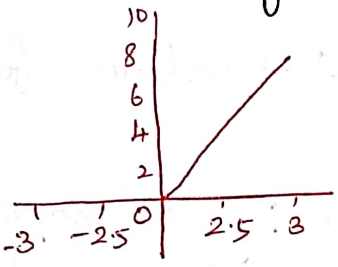
ReLu:- Rectified Linear unit function:-

ReLu activation fn means that if the i/p x is positive, ReLu returns x, if the i/p is negative, it returns 0.

Value range [0, ∞] meaning the function only o/p non-negative values.

Nature:- It is a non-linear activation function, allowing neural n/w to learn complex patterns and making backpropagation more efficient.

Relu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the n/w sparse making it efficient and easy for computation.



$$f(x) = \max(0, x).$$

Tanh function (Hyperbolic Tangent)

Tanh function is very similar to the sigmoid and logistic activation function and even has the same S-shape with the difference in output range of -1 to 1.

Tanh fn is a shifted version of the sigmoid, allowing it to stretch across the y-axis It is defined as.

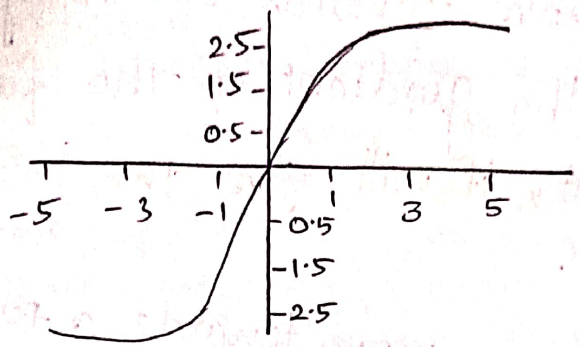
$$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

It can be expressed using the sigmoid function.

$$\tanh(x) = 2 \times \text{sigmoid}(2x) - 1$$

Value Range:- outputs values from -1 to +1

Non linear :- Enable modeling of complex data patterns.



Network Training:-

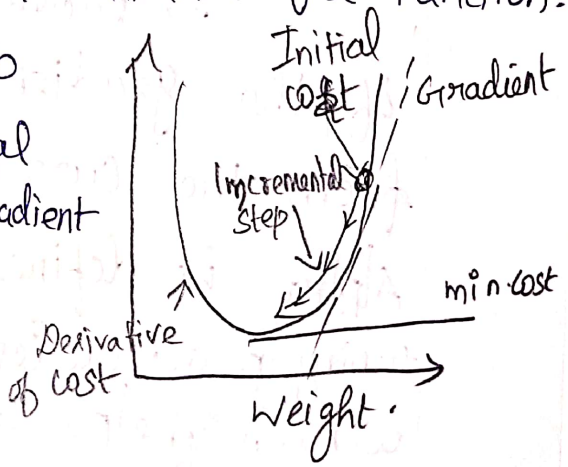
Gradient Descent optimization.

Gradient Descent is known as one of the most commonly used optimization algorithm to train machine learning models by means of minimizing. error blw actual and expected results. Further, gradient descent is also used to train Neural Network.

The main objective of gradient descent is to minimize the convex function using iteration. of parameter updates. once this ML models are optimized, these models can be used as powerful tools for AI and various computer science application

Gradient Descent is defined as one of the most commonly used iterative optimization. algorithms of ML to train the ML and DL models. It helps in finding the local minimum of a function.

The best way to define the local minimum or local maximum of a function using gradient descent is as follows.



If we move towards a -ve or away from the gradient of the current point, it will give the local minimum of that function.

Whenever we move towards a positive gradient or towards the gradient of the function at the current point, we will get the local maximum of that function.

Gradient Ascent :-

This entire procedure is known as gradient Ascent, which is also known as steepest descent. The main objective of using a gradient descent algm is to minimize the cost function using iteration.

To achieve this goal, it performs two steps :-

→ Calculate the first-order derivative of the function to compare the gradient or slope of that function.

→ Move away from the direction of the gradient which means slope increased from the current point by alpha times, where Alpha is defined as learning rate. It is a tuning parameter in the optimization process which helps to decide the length of the steps.

Cost function:-

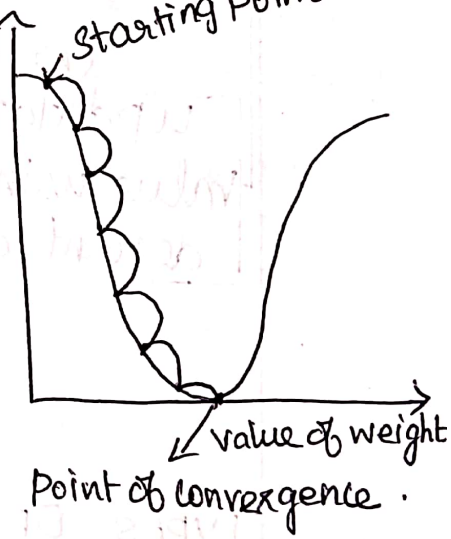
The cost function is defined as measurement of difference or error between actual values and expected values at the current position and present in the form of a single real number.

Working of Gradient Descent:-

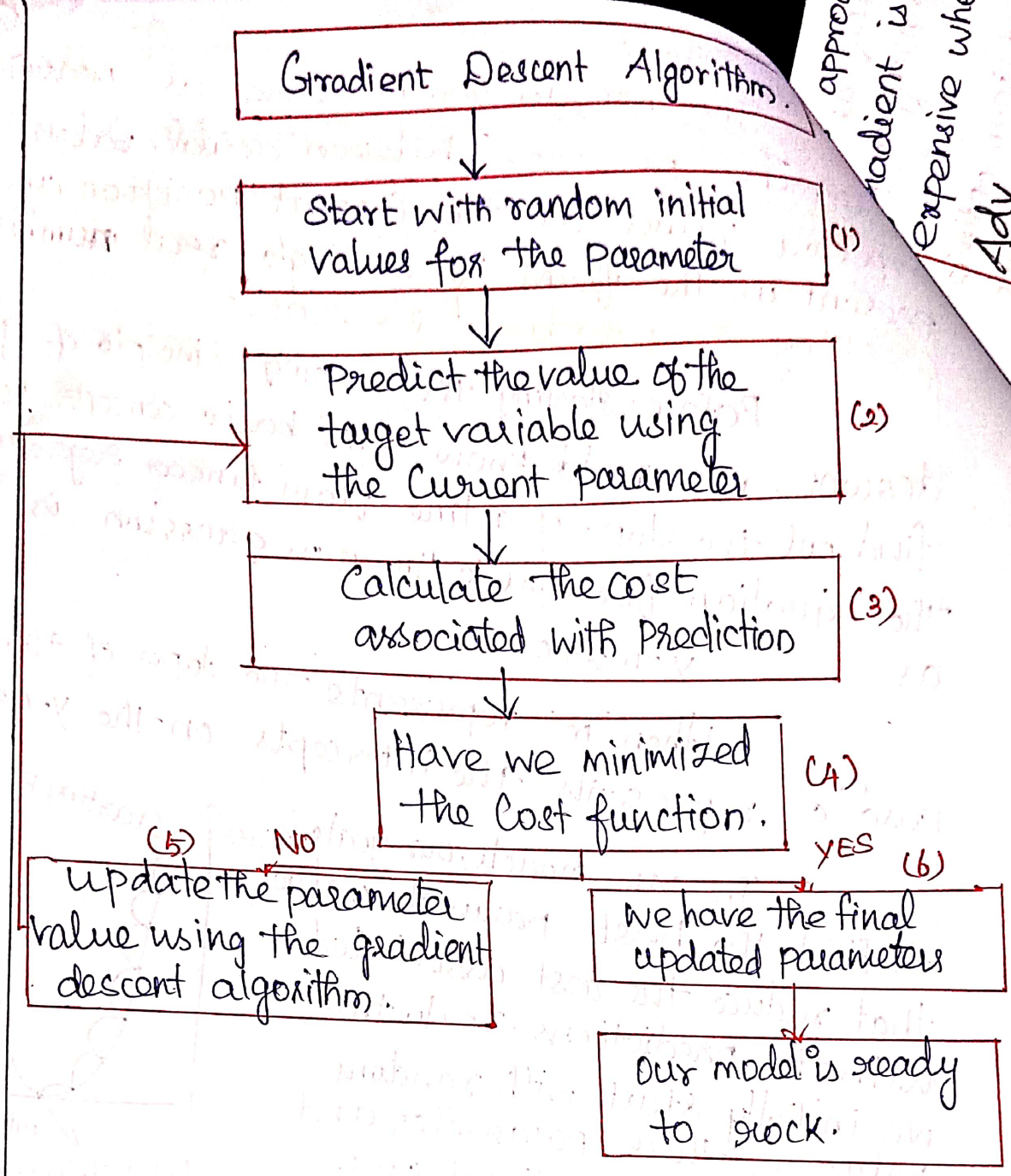
Before starting the working principle of gradient descent, we should know some basic concepts to find out the slope of a line from linear regression. The equation for simple linear regression is given as $y = mx + c$

When 'm' represents the slope of the line, and 'c' represents the intercepts on the y-axis.

In ML models, our goals is to find the best parameter values that reduce the cost associated with the predictions. To do this, we initially start with random values of these parameter and try to find the optimal ones. To find optimal values.



We use the gradient descent algorithm.



TYPES OF GRADIENT DESCENT:-

Based on the error in various training models, the Gradient descent learning algorithm can be divided into Batch gradient descent Stochastic Gradient descent, and mini-batch gradient descent.

1) Batch Gradient Descent:-

Batch gradient descent computes the descent gradient of the cost function using the entire training dataset for each iteration.

is approach ensures that the computed gradient is precise but it can be computationally expensive when dealing with very large datasets.

Adv
- → Accurate Gradient estimates.
- → Good for smooth error surfaces.

DisAdv:
- Slow convergence
- High memory usage
- Inefficient for large datasets.

2) Mini Batch Gradient Descent:-

Mini Batch Gradient Descent combines concepts from both batch gradient descent and Stochastic gradient descent. It splits the training dataset into small batch sizes and perform updates on each of these batches. This approach strikes a balance btw the computational efficiency of batch gradient descent and the speed of Stochastic gradient descent.

Adv
- Faster Convergence
- Reduced memory usage
- Smoother gradient estimation
- Parallelization & speed.

Dis adv.
- choice of Batch size
- Requires more epochs
- Complexity.

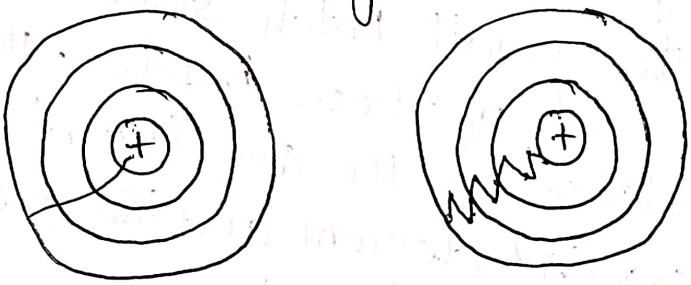
Stochastic Gradient Descent:-

Stochastic gradient descent is an optimization algorithm in machine learning, particularly when dealing with large datasets. It is a variant of the traditional gradient descent algorithm but offers

Several advantages interms of efficiency making it the go-to method for many tasks.

Stochastic Gradient Descent is an optimization algorithm that finds the set of input variables for a target function that results in a minimum value of the target function, Called the minimum of the function.

Stochastic Gradient descent : extension of the gradient descent optimization algorithm for minimizing a loss function of a particular model on a training data set.



Gradient descent Stochastic Gradient Descent.

The word stochastic means random.

In SGD, we pick a few data points randomly (instead of using all data) for each step.

It takes one training example at a time to update the model.

It's faster and needs less memory.

Because it updates often, it may help escape local minimum and reach the global one.

APP of SGD :-

Deep Learning NLP computer vision Reinforcement Learning.

Adv.	DisAdv.
Faster convergence	Noisy Gradient estimates
Lower memory requirements	Convergence issues.
Escape local minima.	Requires Stuffing.

Error Back propagation:-

Back propagation is one of the important concepts of a neural network, our task is to classify our data set. For this, we have to update the weights of parameter and bias. but how can we do that in deep neural network. In the linear regression model, we use gradient descent to optimize the parameter similarly here we also use gradient descent algorithm using Backpropagation

The main feature of Back propagation are the iterative, recursive and efficient method through which it calculates the updated weights to improve the network until it is not able to perform the task for which it is being trained. Derivative of the activation function to be known at network design time is required to Back propagation.

The Backpropagation algorithm look for the minimum value of the error function in weight space using a technique called the data rule or gradient descent. The weights that minimize the error function. is then considered to be a solution to the learning problem. The training algorithm of Back propagation involves four stages which one as follows.

Initialization of weights:-

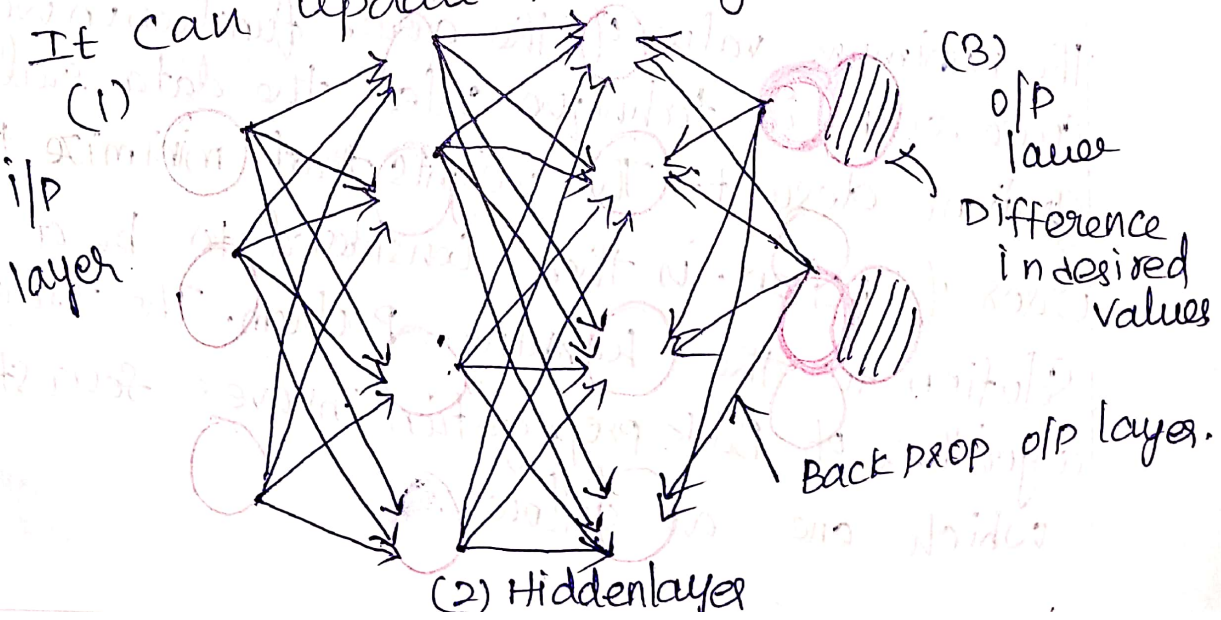
There are some small random are assigned.

Feed forward: Each unit X receives an i/p and transmit this signal to each of the hidden unit $z_1, z_2, \cdots z_n$. Each hidden unit calculate the activation function and sends its signal z_i to each output unit. The o/p unit calculates the activation function to form the response of the give input pattern.

3) Back propagation of errors:-

Each o/p unit, compare activation y_k with the target value T_k to determine the associated error for that unit. It is based on the error, the factor δ_k ($k=1 \cdots m$) is computed and is used to distribute the error at the output unit y_k back to all units in the previous layer similarly the factor δ_j ($j=1 \cdots p$) is computed for each hidden unit z_j.

4) It can update the weight & biases.



Consider the back propagation neural network example diagram to understand.

types of Backpropagation:-

(i) Static Back propagation.

(ii) Recurrent Back propagation.

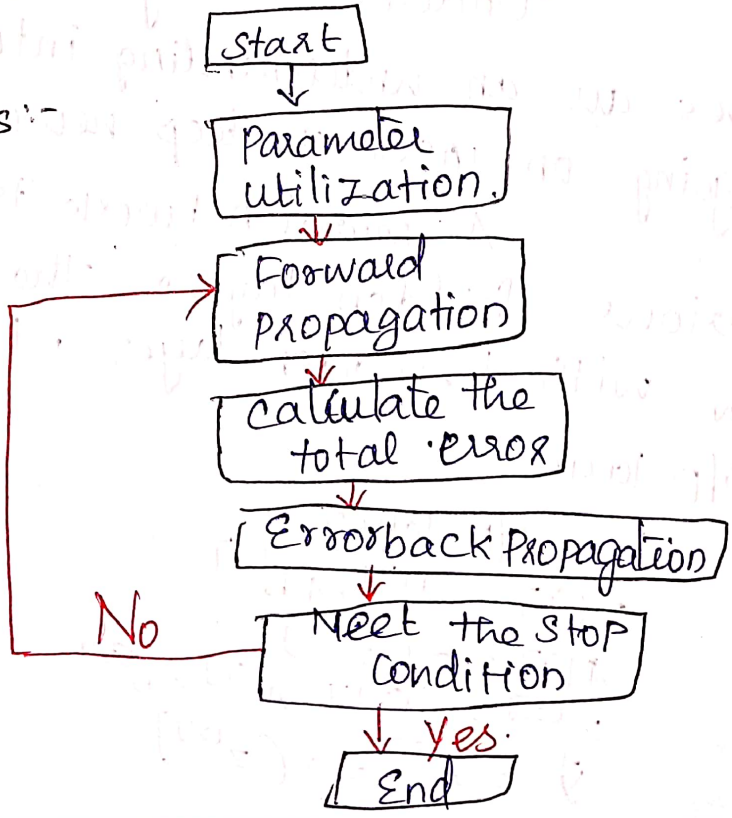
Static Back Propagation:-

In this type of backpropagation the Static output is created because of the mapping of Static input. It is used to resolve Static classification problems like optical character recognition.

Recurrent Back propagation:-

The Recurrent propagation is directed forward or directed until a specific determined value or threshold value is acquired. After the certain values, the error is evaluated and propagated backward.

How it works:-



1) i/p is modeled using real weights w. The weights are usually randomly selected.

2) Calculate the output for every neuron from the input layer, to the hidden layer, to the o/p layer.

(3) Calculate the errors in the outputs.

Error B = Actual o/p - Desired o/p.

(4) Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

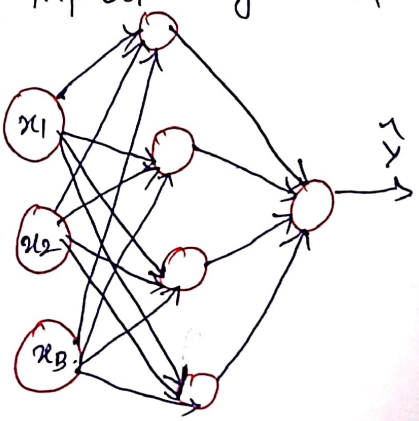
(5) Keep repeating the process until the desired o/p is achieved.

From Shallow Network to Deep Networks :-

Shallow neural networks give us basic idea about deep neural network which consist of only 1 & 2 hidden layer.

Understanding a shallow Neural Network gives us an understanding into what exactly is going on inside a deep neural network.

A neural network is built using various hidden layers. The shallow neural n/w with 1 hidden layer, 1 input layer and 1 o/p layer.



1 - o/p layer

$$z^{[1]} = w^{[1]T} x + b^{[1]}$$

$$A^{[1]} = \sigma(z^{[1]})$$

$$z^{[2]} = w^{[2]T} A^{[1]} + b^{[2]}$$

$$\hat{y} = A^{[2]} = \sigma(z^{[2]})$$

1) The 1st eqn calculates the intermediate output Z[1] of the first hidden layer.

2) The 2nd eqn calculates the final o/p A[1] of the first hidden layer

3) The third equation calculates the intermediate output Z[2] of the output layer.

4) The fourth equation calculates the final output A[2] of the o/p layer which is also the final output of the whole neural network.

Unit Saturation (i.e. the vanishing gradient problem):

The vanishing gradient problem is an issue that sometimes arises when training machine learning algorithm through gradient descent. This most often occurs in neural networks that have several neural layer such as in a deep learning system, but also occurs in recurrent neural networks.

The key point is that the calculated partial derivatives used to compute the gradient as one goes deeper into the network. Since the gradient control how much the network learns during training the gradient are very small or zero, then little to no training can take place, leading to poor predictive performance.

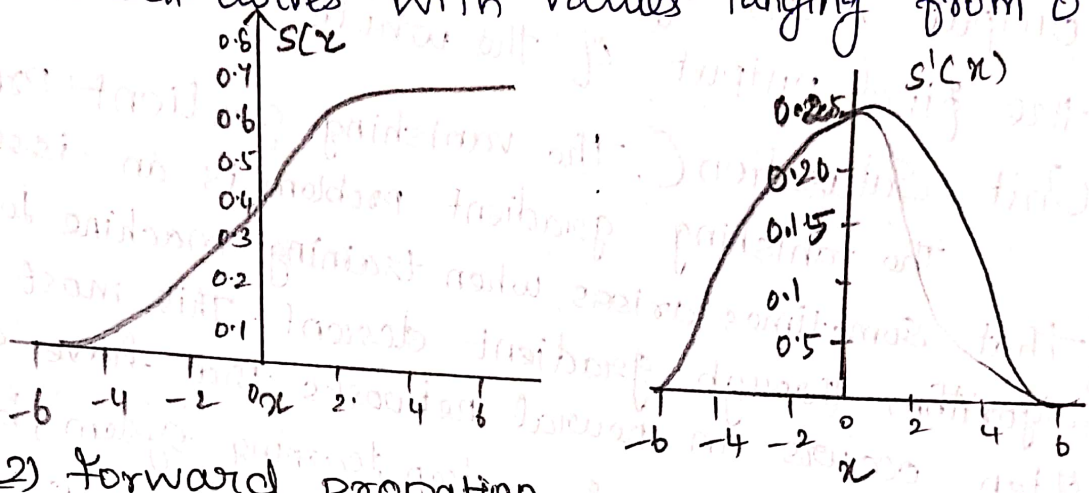
Sigmoid function:-

sigmoid function are used frequently in neural networks to activate neurons. It is a logarithmic function with a characteristics shaps. The o/p value of the functi

is b/w 0 and 1. The sigmoid function for activating the o/p layer is binary classification problems.

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

from the below graph, you can see a comparision b/w the sigmoid function itself and its derivative. First derivatives of sigmoid function are bell curves with values ranging from 0 to 25.



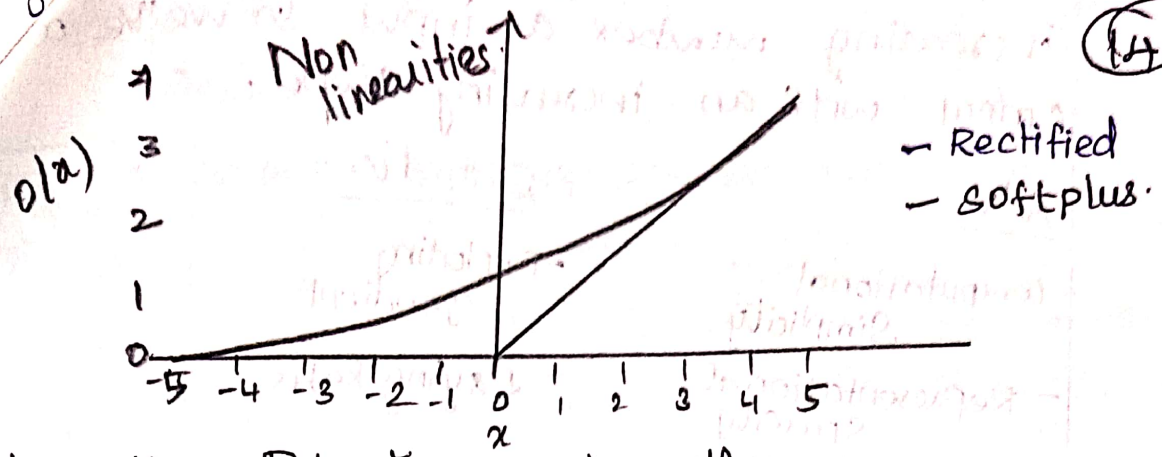
(2) forward propation
(3) Back Propagation

RELU :-

The rectified linear activation unit or ReLu. is one of the few landmarks in the deeplearning revolution. It's simple, yet its far superior to previous activation function like sigmoid or tan h.

$$f(x) = \max(0, x)$$

The diagram below with the blue line is the representation of Rectified linear unit where as the black line is a variant of ReLu includes Leakly ReLu exponential linear unit (ELU) and sigmoid linear unit (SiLU) etc..



Implementing Relu function in python.

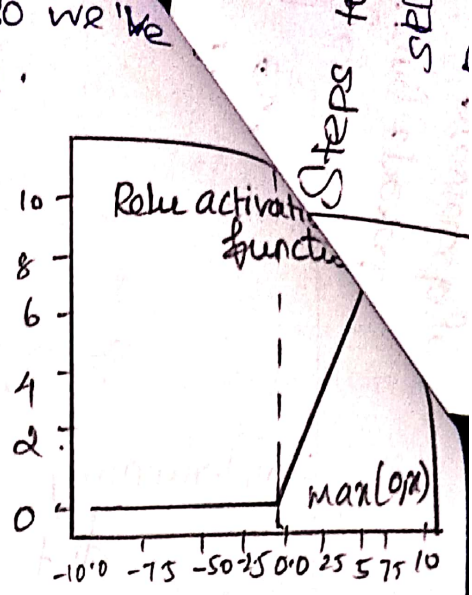
```
def relu(x)
    return max(0.0, x)
```

To test the function, lets run it on a few inputs.

```
x = 1.0
Print ('Applying Relu on (%.1f) gives %.1f' %
                                (x, relu(x)))
x = -10.0
Print ('Applying Relu on (%.1f) give %.1f'
                                % (x, relu(x)))
x = 0.0
Print ('Applying Relu on (%.1f) gives %.1f' %
                                (x, relu(x)))
x = 15.0
Print ('Applying Relu on (%.1f) gives %.1f' %
                                (x, relu(x)))
x = -20.0
Print ('Applying relu on (%.1f) gives %.1f' %
                                (x, relu(x)))
```

We see from the plot that all the -ve values have been set to zero. and +ve values are returned as it, Note that we have given a set of consecutively

increasing number as input so we've output with an increasing slope.



Adv

- Computational Simplicity
- Representational Sparsity
- Linear behaviour
- Mitigation of Vanishing Gradient
- Faster training.

Disadv

- Exploding gradient
- Dying Relu

Hyperparameter Tuning :-

Hyper parameter in ML are those parameter that are explicitly defined by the user to control the learning process. These hyperparameter are used to improve the learning of the model. and their values are set before starting the learning process of the model.

Hyper parameters are defined as the parameter that are explicitly defined by the user to control the learning process.

Some examples of model hyperparameter include.

- The penalty in logistic regression classifier ie: L1 or L2, regularization.
- The learning rate for training a neural network.
- The C and sigma hyperparameter for support vector machine.
- The k in k-nearest neighbors.

Models can have many hyperparameter and finding the best combination of parameter can be treated as a search problem.

activation function

Steps to perform Hyper parameter tuning:-

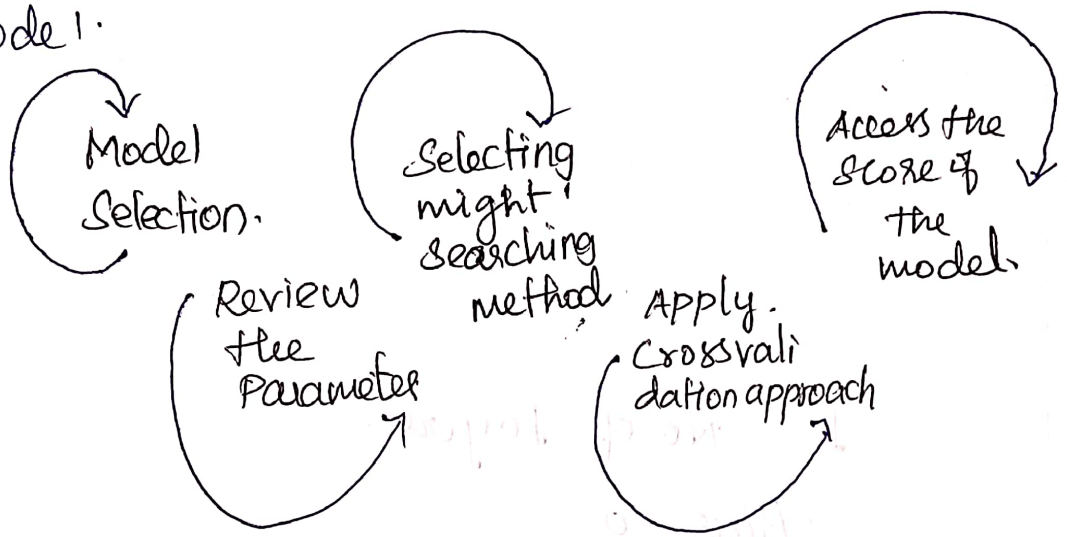
select the right type of model:

Review the list of parameters of the model and build the HP space.

Finding the methods for searching the hyperparameters space.

Applying the cross validation scheme approach.

Access the model score to evaluate the model.



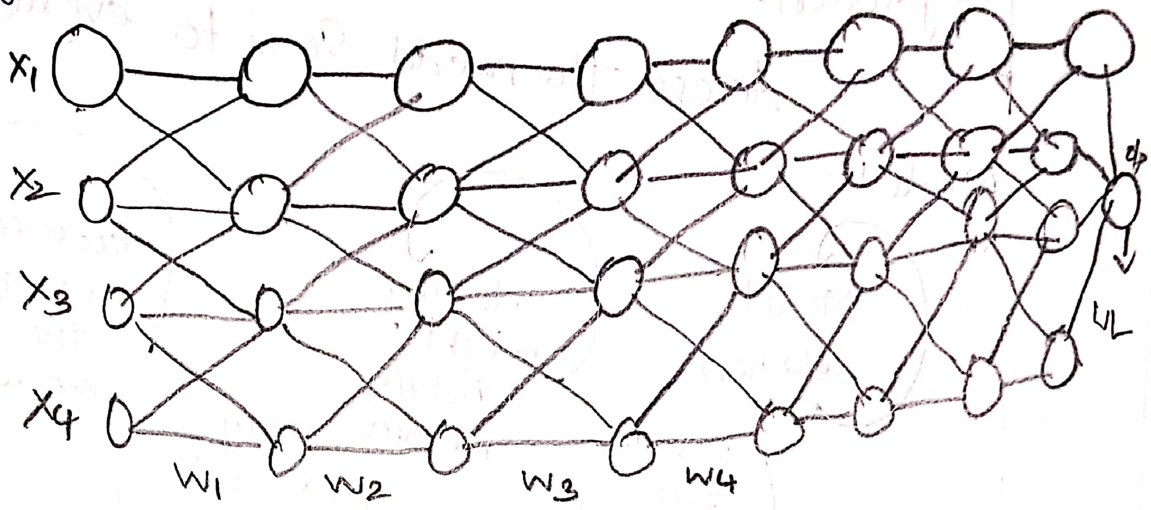
Batch Normalization:-

Normalization is a data preprocessing tool used to bring the numerical data to a common scale without distorting its shape.

Generally when we input data to a machine or deep learning algorithm we tend to change the values to a balanced scale. The reason we normalize parts to ensure that our model can generalize appropriately.

It is a process to makes neural ... faster and more stable through adding ... in a deep neural network. The new layer perform the standardizing and normalizing oper... on the input of a layer coming from a previous la...

A typical Neural N/w is trained using a collected set of i/p data called batch. Similarly the normalizing process in batch normalization takes place in batches, not as a single input.

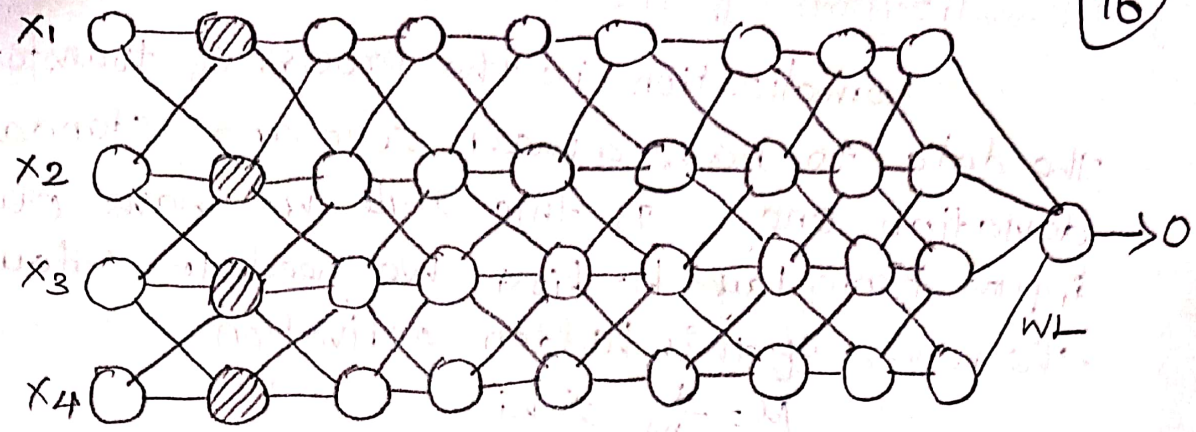


L - No of layers.

Bias - 0

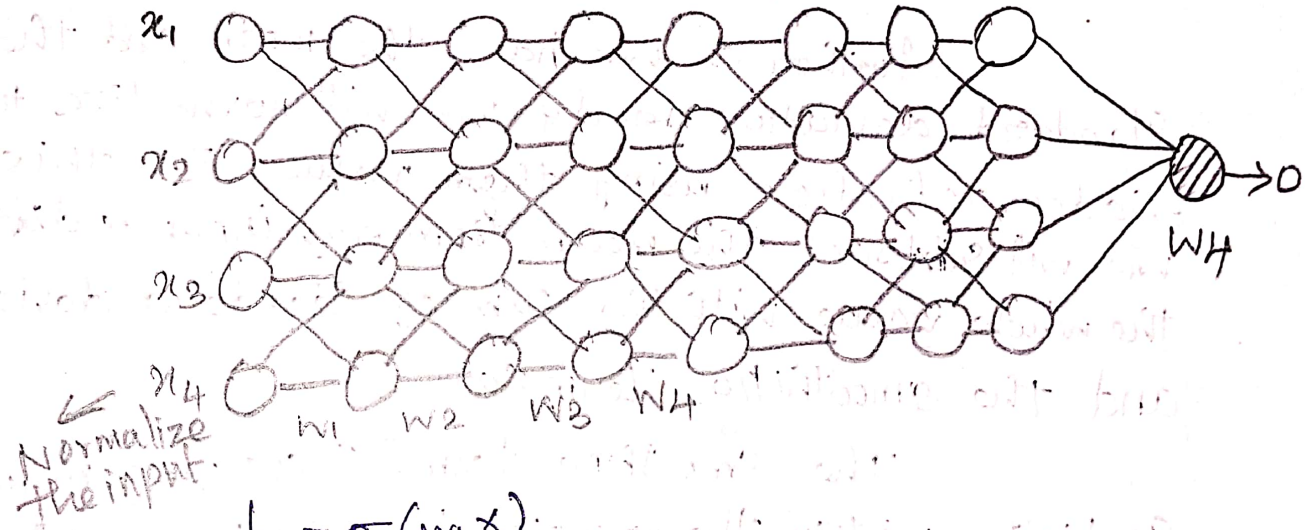
Activation function: Sigmoid.

Initially our i/p, $x_1, x_2, x_3 x_4$ are in normalized form as they are coming from the preprocessing stage, when the i/p passes through the first layer, it transform as a sigmoid function applied over the dot product of i/p x and the weight matrix w.



$$h_1 = \sigma(w_1 x)$$

Similarly, this transformation will take place for the second layer and go till the last layer L as shown in image.



$$h_1 = \sigma(w_1 x)$$

$$h_2 = \sigma(w_2 h_L) = \sigma(w_2 \sigma(w_1 x))$$

$$O = \sigma(w_L h_{L-1})$$

Working of Batch Normalization:-

Since by now we have a clear idea of why we need Batch normalization, Lets understand how it works. It is a two step process. First the input is normalized and later rescaling and offsetting is performed.

Normalization of the input:-

Normalization is the process of transforming the data to have a mean zero and standard deviation one. In this style we have our batch input from layer h. First we need to calculate the mean of this hidden activation.

$$\mu = \frac{1}{m} \sum hi$$

Here m is the no of neuron at layer h. once we have meant at our end, the next step is to calculate the standard deviation of hidden activation.

$$\mu = \sqrt{1/m \sum (hi - \mu)^2}$$

Further as we have the mean and the Standard deviation ready we will normalize the hidden activation using these value. For this we will subtract the mean from each input & divide the whole value with the sum of Standard deviation and the smoothing term (ε)

The Smoothing term (ε) assure numerical Stability within the operation by stopping a division by a zero value.

$$h(i(num)) = \frac{(hi - \mu)}{\sigma + \varepsilon}$$

Adv of Batch Normalization:-

*Speed up the training:-

By Normalizing the hidden layer activation the Batch normalization Speed up the training Process.

* Handle internal covariate shift
* internal covariate shift
* Smoothens the loss function.

Regularization:-

Regularisation refers to techniques that are used to calibrate ML models in order to minimize the adjusted loss function and prevent overfitting or underfitting.

over fitting. Good fitting.

Regularization techniques:-

The commonly used regularization techniques are:

1) L1 regularization
2) L2 regularization
3) Dropout regularization.

A regression model which uses L1 regularization technique is called LASSO regression.

A regression model that uses L2 regularization technique is called Ridge regression.

Lasso regression adds absolute value of magnitude of coefficient as penalty term to the loss function (L).

Working of Regularization

Regularization works by adding a penalty or complexity term of the complex model.

Let's consider the simple linear regression equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots \beta_n x_n + b$$

In the above eqn, 'y' represents the value to be Predicted.

$x_1, x_2 \cdots x_n$ are features for y.

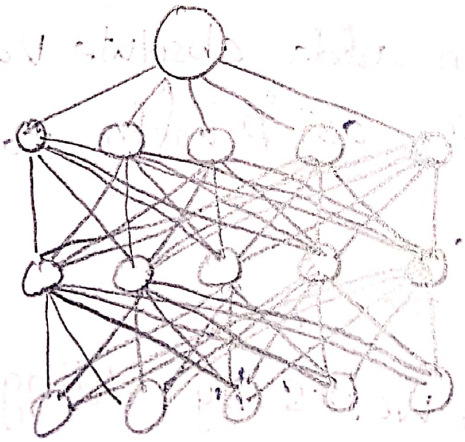
$\beta_0, \beta_1 \cdots \beta_n$ are weights or magnitude attached to the features, respectively.

The Here represents to the bias of model, and b represent the interrupt.

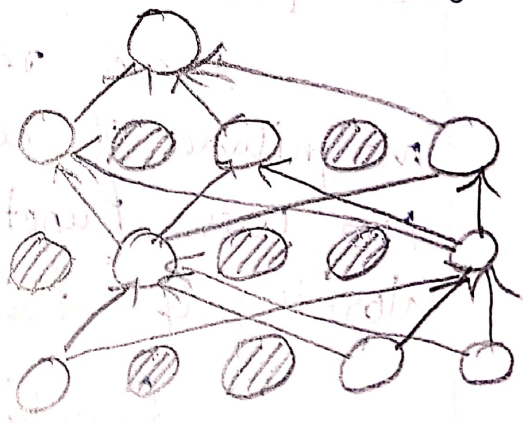
Dropout:-

Dropout in ML refer to the process of randomly ignoring certain nodes in a layer during training.

In the figure below, the neural network on the left represents a typical neural n/w ~~on the left repres~~ where all units are activated. on the right, the red units have been dropped out of the model - The values of their weights and bias are not considered during training.



Standard neural n/w



After applying dropout.

Dropout is used as a regularization technique. it presents overfitting by ensuring that no units are co dependent.

Other Common Regularization Methods:-

1) Early stopping.

Stop training automatically when a specific performance measure (eg: validation loss, accuracy) stops improving.

2) weight decay:-

The n/w to use smaller weights by adding a penalty to the loss function (this ensure that the norms of weights are relatively evenly distributed among it all the weights from heavily influencing n/w output).