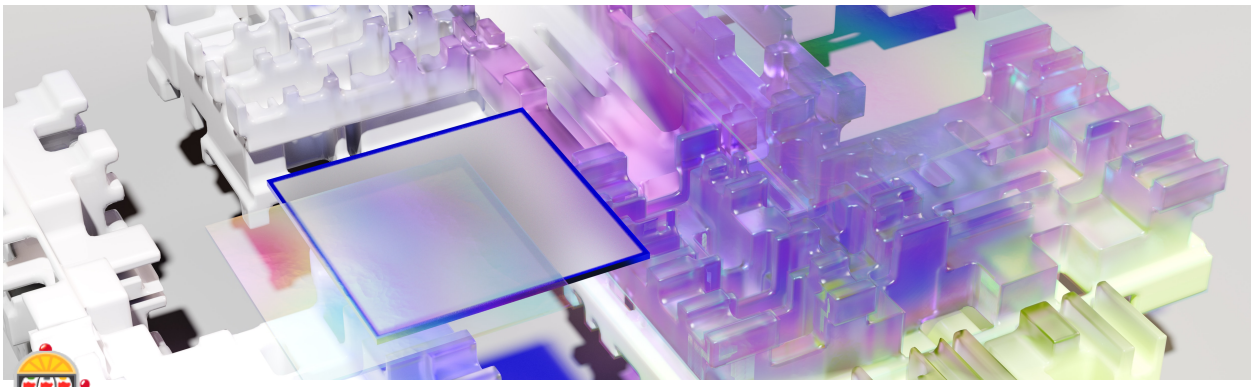
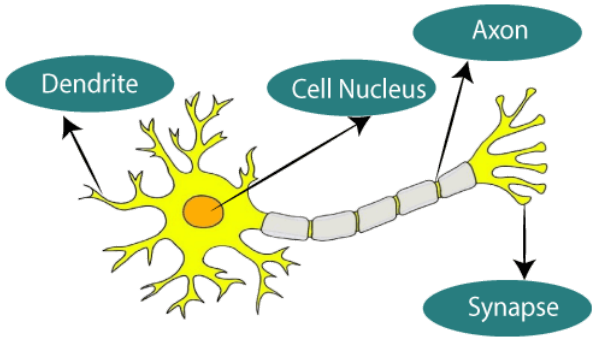
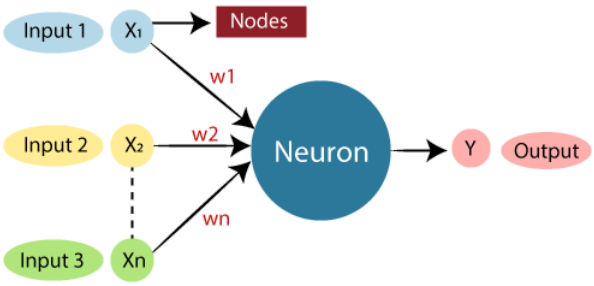


ARTIFICIAL NEURAL NETWORKS CHEATSHEET

ARTIFICIAL NEURAL NETWORKS

- It refers to a biologically inspired sub-field of artificial intelligence modeled after the brain.
- It is computational network based on biological neural network that construct the structure of human brain.
- Similar to human brain that has neurons connected to each other , ANN also has neurons that are connected to one another in various layers of the networks , these neurons are known as nodes.





Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

$$b_{j.std} = b_j(s_x * s_y^{-1}).$$

MACHINE LEARNING

- Field of study that gives a computer the capability to learn without being explicitly programmed.
- It gives the computer that makes it more similar to humans : `Ability to Learn`.

> 💡 **“Field of study that gives computers the capability to learn without being explicitly programmed”**

- Usage of Machine learning :
  - Fraud Detection.
  - Predictive Maintenance.
  - Portfolio Optimization.
  - Automated Tasks.

LIFE OF MACHINE LEARNING PLATFORMS

1. Define a question.
2. Collect data.
3. Visualize data.
4. Train Algorithm.
5. Test the Algorithm.
6. Collect Feedback.
7. Refine the algorithm.
8. Loop to 4-7 steps until the results are satisfying.
9. Use Model to make prediction.
10. Once the algorithm gets the good at drawing the right conclusion , it applies that knowledge to new sets of data.

TYPES OF MACHINE LEARNING

- Supervised Machine Learning.
- Unsupervised Machine Learning.

SUPERVISED MACHINE LEARNING

- It uses training data and feedbacks from humans to learn the relationship between given input and output.
- There are 2 categories of supervised learning :
  - Classification Task.
  - Regression Task.

CLASSIFICATION

- It is the task of “classifying the things” into sub category and it is the type of supervised machine learning algorithm:

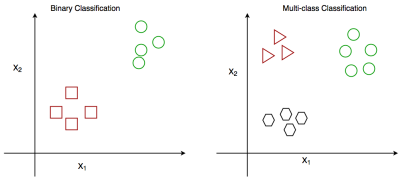
BINARY CLASSIFICATION

- When we have to categorize given data into 2 distinct classes :

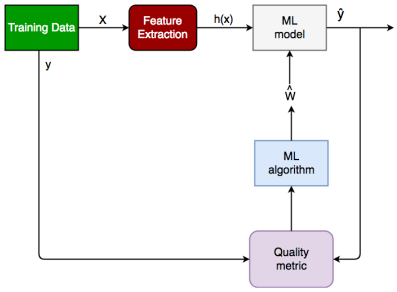
- For example : Suppose on the basis of given health conditions of the person , we have to determine whether the person has the certain disease or not.

MULTICLASS CLASSFICATION

- The number of classes is more than 2.
- For example : On the basis of data about different species of flowers , we have to determine to which specie our observation belongs.



- The following is the generalized diagram of the classification task.



TYPES OF CLASSIFIER

- Linear Classifier.
- Tree-Based Classifier.
- Support Vector Machine.
- Artificial Neural Networks.
- Bayesian Regressions.
- Gaussian Naive Bayes Classifier and etc.

PRACTICAL APPLICATION OF CLASSIFICATION

- Google’s self driving can uses deep learning-enabled classification techniques which enables it to detect and classify obstacles.
- Spam E-mail filtering is one of the most widespread and well - recognized uses of classification techniques.

DATA MINING

- It means digging deep into data and the purpose of digging is to gain patterns and to gain the knowledge of the patterns.
- In the process of data mining :
  - Large data is sorted first.
  - Then the patterns are identified and relationships are established to perform data analysis and solve problem.

REGRESSION ANALYSIS

- It is an statistical process for estimating the relationship between the dependent variables and one or more independent variable or predictors.
- It explains the changes in criteria in relation to changes in select predictors.

TYPES OF REGRESSIONS

- Linear Regressions
  - It is used for predictive analysis.
  - It is linear approach for modeling the relationship between the criterion or the scale response and the multiple predictors or explanatory variables.
- Polynomial Regression
  - It is used for curvilinear data.
  - Polynomial regression is fit with the method of least square.
  - The goal of the regression analysis is to model the expected value of a dependent variable y in regards to the independent variable x . The equation for polynomial regressions is :

$$l = \beta_0 + \beta_0 x_1 + \epsilon.$$

- Stepwise Regression
  - It is used for fitting regression models with predictive models.
  - It is carried out automatically.
  - With , each step the variable is added or substracted from the set of explanatory variable.

$$b_{j.std} = b_j(s_x * s_y^{-1}).$$

- Ridge Regressions
  - It is a technique for analyzing multiple regression data.
  - When multicollinearity occurs , least square estimates are unbiased.

$$\beta = (X^T X + \lambda * I)^{-1} X^T y.$$

- Lasso Regressions
  - It performs both variable selection and regularization.
  - It used the soft thresholding.

$$N^{-1} \sum_{i=1}^{N} f(x_i, y_I, \alpha, \beta).$$

- Elastic Net Regression

- It linearly combines the penalties of the lasso and ridge methods.
- It is used to support vector machine.

$$||\beta||_1 = \sum_{j=1}^{p} |\beta_j|.$$

UNSUPERVISED MACHINE LEARNING

- It is the training of a machine using information that is neither classified not labelled and allowing the algorithm to act on the information without guidance.
- No training will be given to the machine , as there are no teacher are present.
- Unsupervised machine learning is classified into 2 major categories :
  - Clustering.
  - Association.

CLUSTERING

- It is where you want to discover the inherent groupings in the data.
- such as grouping customers by purchasing behavior.
- Types of clustering :
  - Hierarchical Clustering.
  - K-means Clustering.
  - Principal Component Analysis.
  - Single Value Decomposition.
  - Independent Component Analysis.

ASSOCIATION

- It is used where you want to discover rules that large portions of your data.
- such as people that buy X also tend to buy Y.

DIFFERNECE BETWEEN SUPERVISED AND UNSUPERVISED MACHINE LEARNING ALGORITHMS

Parameters	Supervised machine learning	Unsupervised machine learning
Input Data	Algorithms are trained using labeled data.	Algorithms are used against data that is not labeled
Computational Complexity	Simpler method	Computationally complex
Accuracy	Highly accurate	Less accurate
No. of classes	No. of classes is known	No. of classes is not known
Data Analysis	Uses offline analysis	Uses real-time analysis of data
Algorithms used	Linear and Logistics regression, Random forest, Support Vector Machine, Neural Network, etc.	K-Means clustering, Hierarchical clustering, Apriori algorithm, etc.

REINFORCEMENT LEARNING

- It is about taking suitable action to maximize reward in a particular situation.
- It is used by various software and machine which is used to find the best possible behavior or path it should take in a specific situation.

MAIN POINTS OF REINFORCEMENT LEARNING

- Input : The input should be an initial state from which the model will start.
- Output : There are many possible outputs as there are variety of solutions to a particular problem.
- Training : The training is based on the input , The model will return to the state and the user will decide to reward or punish the model based upon its input.
- The model keeps on continue to learn.
- The best solutions is decided based upon the maximum reward.

TYPES OF REINFORCEMENTS

- There are 2 types of Reinforcements :
  - Positive Reinforcements.
  - Negative Reinforcements.

POSITIVE REINFORCEMENTS

- It is defined as when the event occurs due to a particular behavior.
- It increases the strength and the frequency of the behavior.
- Advantages of the Reinforcement Learning :
  - Maximize the performance.
  - Sustain changes for a longer period of time.
  - Too much reinforcement can lead to an overload of state which can diminish the results

NEGATIVE REINFORCEMENTS

- Strengthening of behavior because an negative condition is stopped or avoided.
- Advantages of Reinforcement Learning :
  - Increases Behavior.
  - Provide defiance to a minimum standard of performance.
  - It only provides enough to meet up the minimum behavior.

APPLICATIONS OF REINFORCEMENT LEARNING

- It is used in robotics for industrial automation.
- It is used in machine learning for data processing.
- It is used in creation of training systems that provide custom instructions and materials according to the need of the students.

SITUATIONS IT CAN HANDLE THE BEST

- A model of the environment is known , but the analytic solution is not available.
- Only the simulation model of the environment is given (subject of the simulation-based optimization)
- The only way to collect information about the environment is to interact with it.

MACHINE LEARNING PROCESS

- The machine learning is an phenomenal task which makes machine capable of understanding and make it suitable enough to predict the desirable output for the given input.
- There are 5 major steps in the machine learning process:
    - Data Acquisition
    - Data Cleaning.
    - Model Training.
    - Model Testing.
    - Model Deployment.

DATA ACQUISITION

- It is the process to get the data.
- It will depend on the type of data you gather and source of data.
- Now this data can be from :
    - Static data from existing database.
    - Real time data from IOT system or software.
    - Data from other repository.

DATA CLEANING

- Real world data is often unorganized , redundant or has missing elements.
- In order to feed that data into the machine learning model , first we will need to clean that data and this will involve the crucial steps.
- More will be the clean data , more accurate will be the model.
- Data can be in the format - CSV , XML , JSON etc.
- After cleaning the data should be in the correct format that can be fed into the machine learning platforms.
- After the data cleaning the dataset is divided into the testing and training database.
    - Training dataset is used to train the model.
    - Testing dataset is used to test the model.
- Here are some more things to keep in the mind while splitting the dataset into training and testing:
    - Split range is usually 20% to 80% between the training and testing stages.
    - You cannot mix or reuse the same data for the training and testing purposes.
    - Using the same data or both dataset can result in the faulty model.

EXPLORATORY DATA ANALYSIS

- Finding all the variables and understanding them.
- Importing a dataset.
- Using the head command.
- Using shape for rows and columns.
- check whether the data is balanced or not.
- Check no of samples are same or not.
    - This you can do by the syntax value __counts()

UNIVARIATE ANALYSIS

- Uni means single and variate means variation.
    - Histogram is used to univariate.
    - Histogram is continuous.
- For the outliar dataset we will be using the box-plot.
- Seaborn will be used for visualization process.
- We can plot pairplots.

FITTING OF MISSING VALUES

- SINGLE IMPUTATIONS
    - Fill with mean and median of the columns.
- MULTIPLE IMPUTATIONS
    - Model have missing value with what your model finds.
- KNN
    - Fill data with value from another example that is similar.
    - It is instance based ML Model.

MODEL TRAINING

- A Machine learning algorithm is used to train the model.
- These algorithm leverages mathematical modelling to learn and predict the behaviors.
- These algorithms fall under the 3 categories :
    - Binary.
    - Classification.
    - Regression.

MODEL TESTING

- After the model is trained , now its time to test and validate for the further processing.
- Now here by testing dataset we will check the accuracy of the model.
- It the result are not satisfactory , model should be further improved.
- Things which we can do to refine and improve the model :

- Review the model with the business stakeholders and take in their inputs.
- Reconsider the algorithm which you have chosen to train the model.
- Adjust the parameters of the algorithm which you have chosen.

DATA PREDICTION

- It gives output after it has been trained on historical dataset and applied to new data when forecasting the like hood of particular outcome.
- For example : Stock Prediction in real life.

DATA VISUALIZATION

- It will make the correlation matrix and collect data from past experience and predict possibilities of dataset.

DEPLOYMENT

- Once the model is trained , deploy and pipeline it to production for application consumption.

Which Algorithm we should choose ?
- Supervised Machine Learning algorithms are used if you want to train the model for prediction or classification. For example : identifying the cars from the web footage , predicting stock prices etc.

- Unsupervised learning algorithms can be used if you want to explore the data that you have and find the good internal representation. For example , splitting a dataset into clusters.

ORIGIN OF NEURAL NETWORK

1943 : Warren McCulloch & Walter Pitts

- Wrote paper on how neuron actually works.
- Modeled simple neural network with electrical circuits.

1949 : Donald Hebb

- It pointed out neural pathways are strengthened each time they used in his book `Organization of Behaviour`

1950 : Nathanial Rochester

- First person from IBM laboratory to simulate a neural network

1957 : John Von Neumann

- Suggested imitating simple neuron functions by using telegraph relays and vacuum tubes.

1959 : Bernard Widrow & Marcian Hoff

- Developed first Neural Network applied in real world problem called `ADALINE` and `MADALINE`.

1969 : Marvin Minsky & Seymour Papert

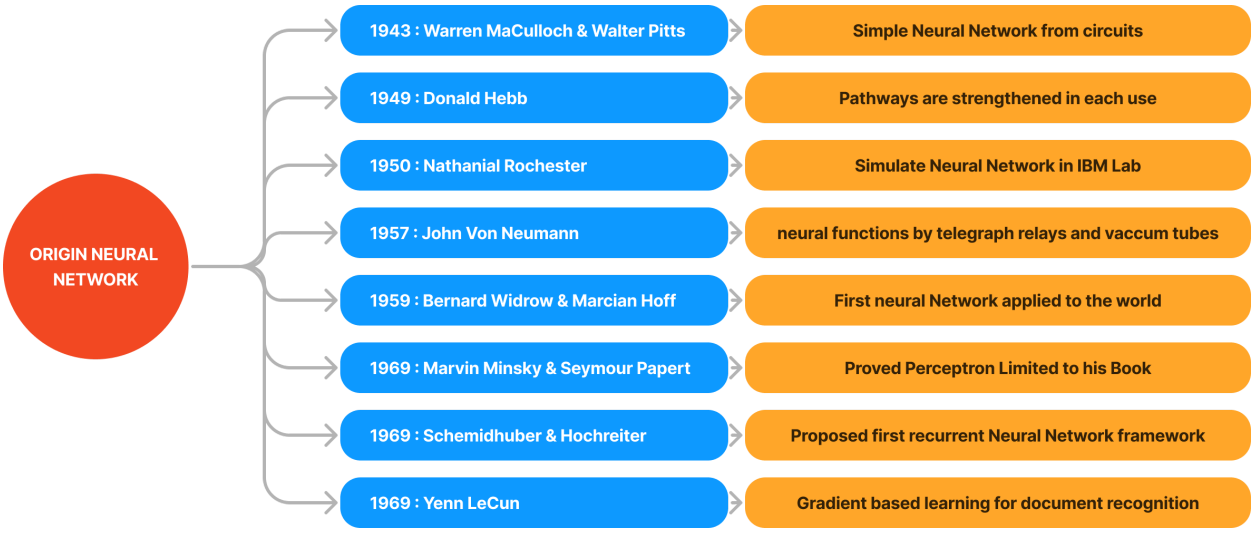
- Proved that perceptron to be limited to their book `Perceptron`.

1997 : Schemidhuber & Hochreiter

- Proposed recurrent neural network framework LSTM.

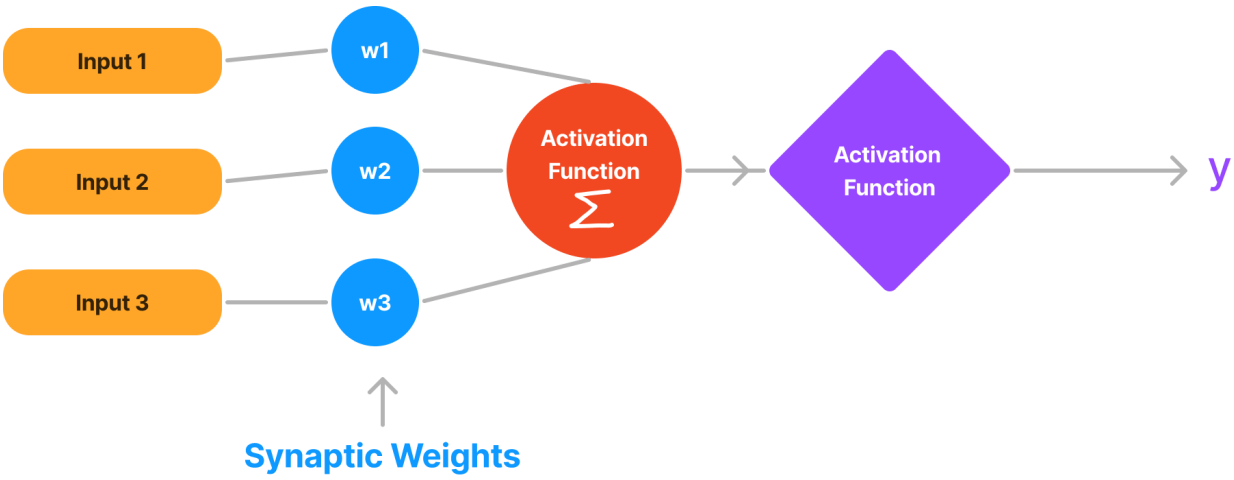
1998 : Yann LeCun

- Published gradient based learning applied to document recognition.



NEURAL NETWORK

- Workhouses of deep Learning.
- They also try to make good prediction.
- Neural Networks are multilayer network of neurons that , we use to classify things and make predictions , etc.



- First Layer = Yellow Color = Input Layer
- First Hidden Layer = Blue = Layer of Neurons.
- The output of activation function is purple in color , it represents output layer.
- Arrow represents how all the neurons are interconnected and how data travels from the input layer to the output layer.

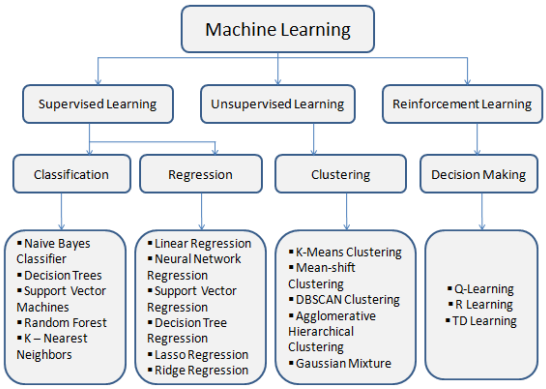
PICTURE OF NEURAL NETWORK

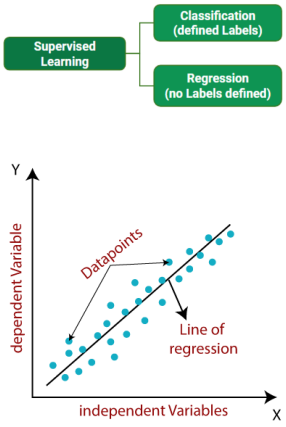
$$\text{Sigmoid}(B1 \cdot X + Bo) \quad \text{Predicted Probability}$$

- X = lone feature we give to model in order to calculate prediction.
- B1 = Estimated Slope Parameter for our logistic regression , Tells how much Log_Odds changes as X changes.
- Bo = Similar to intercept term from regression.
- The blue neuron also include sigmoid activation function.
- Then we will get the predicted probability by applying the sigmoid function to quantity.

LEARNING AND ITS TYPE

- Learning means to do and adapt the change in itself as and when there is a change in the environment.
- There are 3 types of ANN :
  - Supervised.
  - Unsupervised.
  - Reinforcement.





SUPERVISED LEARNING

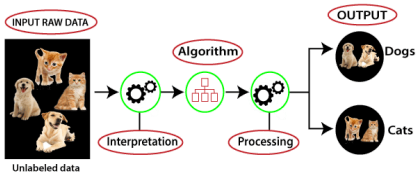
- Model is getting trained on labelled dataset.
- A Labelled dataset is one that has both input and output parameters.
- Here both training and validation , datasets are labelled.

REGRESSION

- Supervised Learning Task.
- Where output is having continuous task.

UNSUPERVISED LEARNING

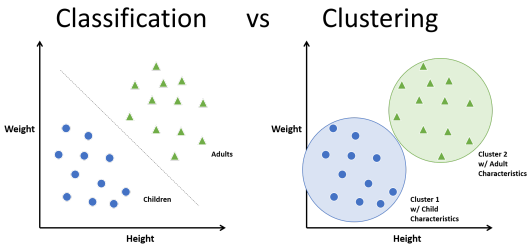
- Using AI algorithms to identify patterns in dataset containing data points that are neither classified nor labeled.

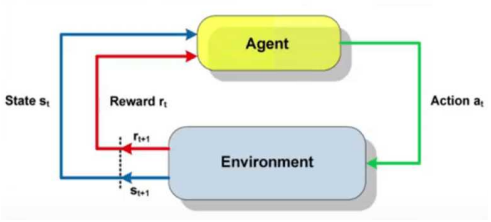


- Unsupervised is further classified as “Clustering”.

CLUSTERING

- Task of dividing the population or data points into number of groups such that data points in the same groups are more similar to other data points in the same group then those in the other group.





REINFORCEMENT LEARNING

- Machine learning trained method based on rewarding desired behaviors and/or punishing undesired ones.
- Reinforcement learning is able to perceive and interpret its environment , take actions and learning through trail error.

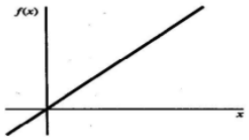
ACTIVATION FUNCTION

- It is used to decide whether the neuron should be activated or not.
- There are 4 activation functions :
  - Identity functions.
  - Binary step functions.
  - Binary Sigmoid.
  - Bipolar Sigmoid.

IDENTITY FUNCTION

- It collects the input and produces an output which is proportionate to the given input.
- It gives multiple outputs not just the True an False.

f(x) = x - for all x



$$f(x) = \begin{cases} 1 & \text{if } x >= \theta \\ 0 & \text{if } x < \theta \end{cases}$$

Binary Step Function ⬆️⬆️

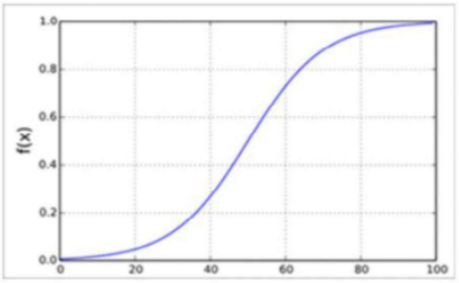
BINARY STEP FUNCTION

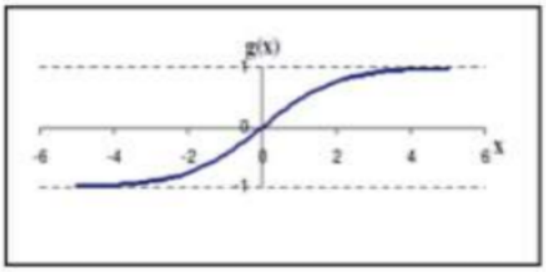
- Binary step function is **a threshold-based activation function** which means after a certain threshold neuron is activated and below the said threshold neuron is deactivated.

BINARY SIGMOID FUNCTION (Logistic Function)

- A sigmoid function is **a bounded, differentiable, real function that is defined for all real input values.**
- Input should **has a non-negative derivative at each point and exactly one inflection point.**

$F(x) = [\ 1/(1+ e^{-ax})]$ --





- Bipolar Sigmoid Function ⬆️⬆️

BIPOLAR SIGMOID FUNCTION

- **a type of activation function or “squashing function” that is widely used in Hopfield neural network.**

PROJECT 1

To design McCulloch Pitts / Perceptron Model for the logic gates.

UNDERSTANDING THE PROBLEM STATEMENTS

- We have to implement logic gates like :
  - OR
  - NOT
  - AND etc.

Experiment 1

Below given we have created all the logic gates and using the basic Logic Gates of AND , OR and NOT we have created XOR gates.

In [1]:
```
def ANDlogic(x1,x2,w1=1,w2=1):
    yin = x1*w1 + x2*w2
    if (yin >=2 ):
        return 1
    else:
        return 0
```

In [2]:
```
def ORLogic(x1,x2,w1=1,w2=1):
    yin = x1*w1 + x2*w2
    if(yin==0):
        return 0
    else:
        return 1
```

In [8]:
```
def NOTLogic(x1,w1=1):
    yin = x1*w1
    if(yin==0):
        return 1
    else:
        return 0
```

In [11]:
```
def XORLogic(x1,x2,w1=1):
    output = ((NOTLogic(x1,w1)*x2) + (x1*NOTLogic(x2,w1)))
    print(output)
```

The output of the Logic Gates is given below and it is also verified and you can test this with different weights and as the initial setting I have changed the weights to 1

In [12]:
```
XORLogic(1,1,1)
XORLogic(1,0,1)
XORLogic(0,1,1)
XORLogic(0,0,1)
```

```
0
1
1
0
```

PROJECT 2

To plot the various types of activation functions used in the artificial neural networks

UNDERSTANDING THE PROBLEM STATEMENTS

- Project is used for implementing various types of the activation functions.
  - Binary Functions.
  - Sigmoid Functions.

- Bipolar Functions.
- ReLu Functions.
- Leaky ReLu Functions.

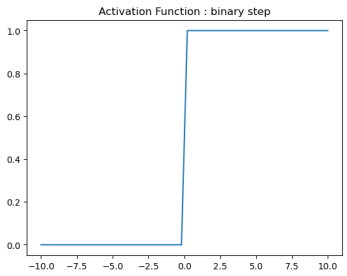
BINARY STEP FUNCTION

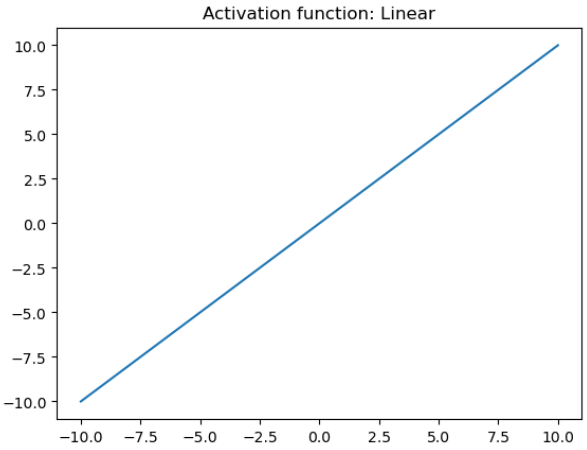
- It returns the value either 0 or 1:
  - It returns ‘0’ If the input is the less then 0.
  - It returns ‘1’ If the input is greater then 0.

```
def binaryStep(x):
    return np.heaviside(x,1)
```

- Plotting Code is given below:

```
x = np.linspace(-10,10)
plt.plot(x,binaryStep(x))
plt.axis('tight')
plt.title('Activation Function : binary step')
plt.show()
```





LINEAR ACTIVATION FUNCTION

- It simply returns the input back.
- The code of the following is given below:

```
def linear(x):
    return x
```

- Plotting code is given below:

```
x = np.linspace(-10,10)
plt.plot(x,linear(x))
```

```
plt.axis('tight')
plt.title('Activation function: Linear')
plt.show()
```

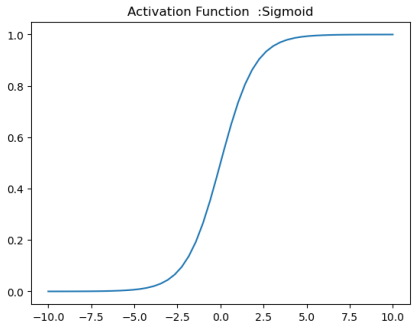
BINARY SIGMOID ACTIVATION FUNCTION

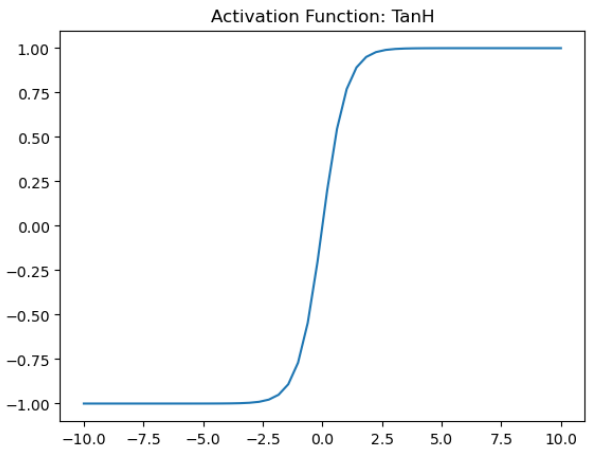
- It returns the value between 0 and 1.
- Activation function in the deep learning network sigmoid function is not considered good since near the boundaries the network doesn't learn quickly.
- It is because gradient is almost zero near the boundaries.
- The code of the following is given below:

```
def sigmoid(x):
    return 1/(1+np.exp(-x))
```

- The code of the Plotting is given below:

```
x = np.linspace(-10,10)
plt.plot(x,sigmoid(x))
plt.axis('tight')
plt.title('Activation Function  :Sigmoid')
plt.show()
```





TANH ACTIVATION FUNCTION

- It is an non linear activation function.
- Its output ranges between -1 and 1.
- It suffers from gradient problem near the boundaries such as sigmoid function does.
- The code of the tanh function is given below:

```
def tanh(x):
    #return the value (1-exp(-2x))/(1+exp(-2x)) which lies in the range -1 and 1
    return np.tanh(x)
```

- The plotting code is given below :

```
x = np.linspace(-10,10)
plt.plot(x,tanh(x))
plt.axis('tight')
plt.title('Activation Function: TanH')
plt.show()
```

RELU ACTIVATION FUNCTION

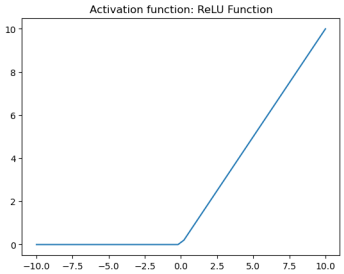
- It is used in the deep learning networks.
- It is less computational expensive then other non linear activation function.
  - It return 0 if the input is less then 0.
  - It returns x if the input is greater then 0
- The code for the same is mentioned below:

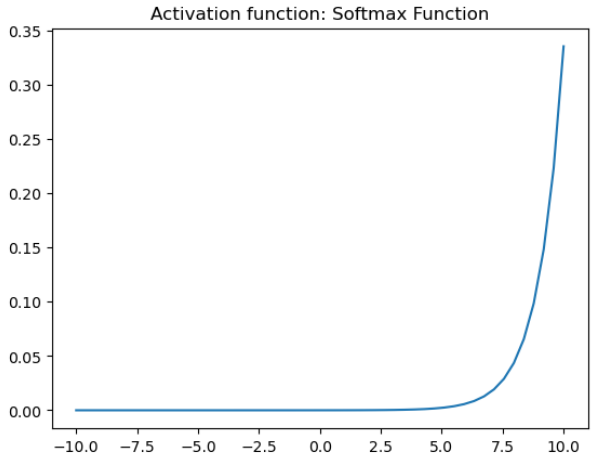
```
def RELU(x):
    x1=[]
    for i in x:
        if i<0:
            x1.append(0)
        else:
            x1.append(i)
    return x1
```

If you change the i with the 1 in the above code you get the same graph as the binary step function.

- The plotting code of the following is given below :

```
x = np.linspace(-10,10)
plt.plot(x,RELU(x))
plt.axis('tight')
plt.title('Activation function: ReLU Function')
plt.show()
```





SOFTMAX ACTIVATION FUNCTION

- It turns the logits , the numerical output of the last linear layer of a multi-class classification neural network into probabilities.
- We code to implement the Soft activation function as given below :

$$\text{Softmax } \sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

$$\text{Sigmoid } S(x) = \frac{1}{1+e^{-x}}$$

```
def softmax(x):
    return np.exp(x)/np.sum(np.exp(x), axis=0)
```

- The plotting code of the following is given below:

```
x = np.linspace(-10,10)
plt.plot(x, softmax(x))
plt.axis('tight')
plt.title('Activation function: Softmax Function')
plt.show()
```

LEAKY RELU FUNCTION

- It is improved version of the ReLu function.
- It is used to solve the dying problem of the ReLu function.
- It adds the slight slope in the negative range in order to prevent the dying ReLu issue.
- If the input to the function x is positive then the output will be x.
- Otherwise the output will be `ax` where:
  - a is the small magnitude.
- The mathematical Representation of it is given below :

f(x) = max(0.05x,x)

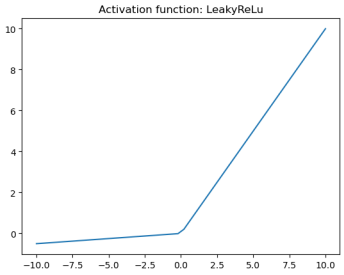
$$f(x) = max(0.05x, x)$$

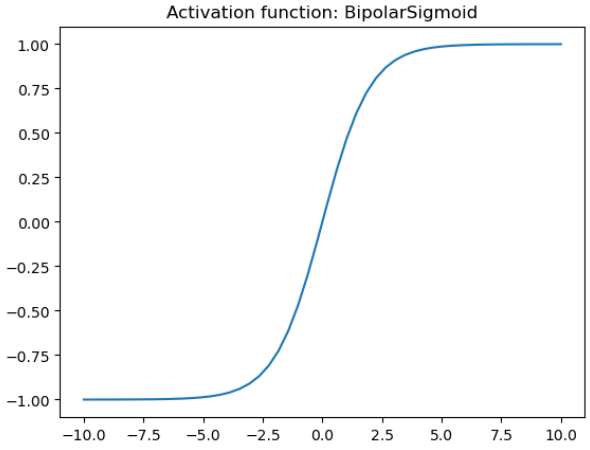
- The code of the following is given below :

```
def leaky_relu(x):
    data = [max(0.05*value, value) for value in x]
    return np.array(data, dtype=float)
```

- The plotting code of the following is given below :

```
x = np.linspace(-10,10)
plt.plot(x,leaky_relu(x))
plt.axis('tight')
plt.title('Activation function: LeakyReLu')
plt.show()
```





BIPOLAR SIGMOID ACTIVATION FUNCTION

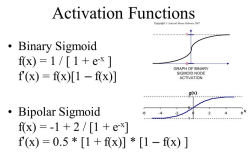
- The code of the following is given below;

```
def bipolarsigmoid(x):
    return (-1) + (2/(1+np.exp(-x)))
```

- The plotting code is given below:

```
x = np.linspace(-10,10)
plt.plot(x,bipolarsigmoid(x))
plt.axis('tight')
```

```
plt.title('Activation function: BipolarSigmoid')
plt.show()
```



PROJECT 3

Implement K Nearest Neighbors Classifier on diabetes dataset

- Download the dataset from given below link.

Pima Indians Diabetes Database

Predict the onset of diabetes based on diagnostic measures

https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

UNDERSTANDING THE PROBLEM STATEMENT

- You have given the diabetes dataset and you have to apply KNN classifier on the Iris dataset.

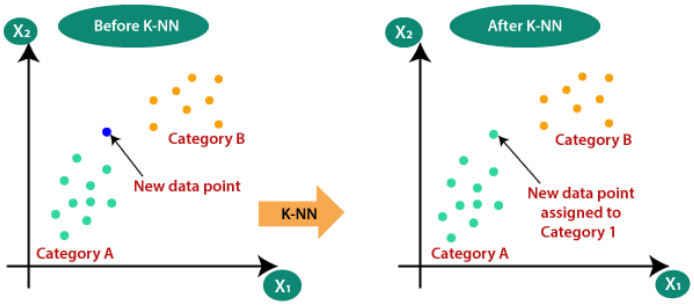
KNN CLASSIFIER

- It is one of the simplest Machine learning algorithm based on supervised machine learning.
- It is non parametric algorithm.
- It is lazy learner algorithm.
    - Because it does not learn from the training set immediately instead it stores the dataset and at the time of clustering it performs actions on the dataset.

HOW DOES KNN WORK ?

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.

- **Step-6:** Our model is ready.



ADVANTAGES OF KNN ALGORITHM

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

DISADVANTAGES OF KNN ALGORITHM

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

CODE

- Data preprocessing Step.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix , ConfusionMatrixDisplay

my_data = pd.read_csv('diabetes.csv')
x = my_data.iloc[:,[0,1,2,3,4,5,6,7]].values
y = my_data['Outcome']
X_train , X_test , y_train, y_test = train_test_split(x,y,test_size=0.25, random_state=43)
```

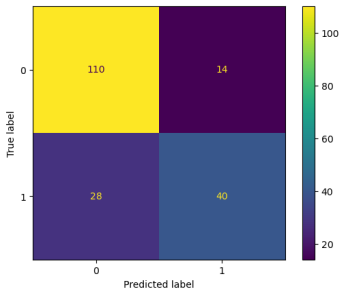
- Using the classifier (In this step we will use the KNN Classifier)

```
Classifier = KNeighborsClassifier(n_neighbors=5, metric="euclidean", p=2)
Classifier.fit(X_train, y_train)
y_pred = Classifier.predict(X_test)
```

- Print the accuracy score and confusion matrix of the following.

```
In [18]: score = accuracy_score(y_test,y_pred)

In [19]: print('Accuracy score is : ',score)
         Accuracy score is :  0.78125

In [27]: cm = confusion_matrix(y_test,y_pred)

In [28]: disp = ConfusionMatrixDisplay(cm)

In [29]: disp.plot()
```

```
score = accuracy_score(y_test,y_pred)
print('Accuracy score is : ',score)
cm = confusion_matrix(y_test,y_pred)
disp = ConfusionMatrixDisplay(cm)
disp.plot()
```



PROJECT 4

Implement Perceptron Classifier on the Diabetes Dataset

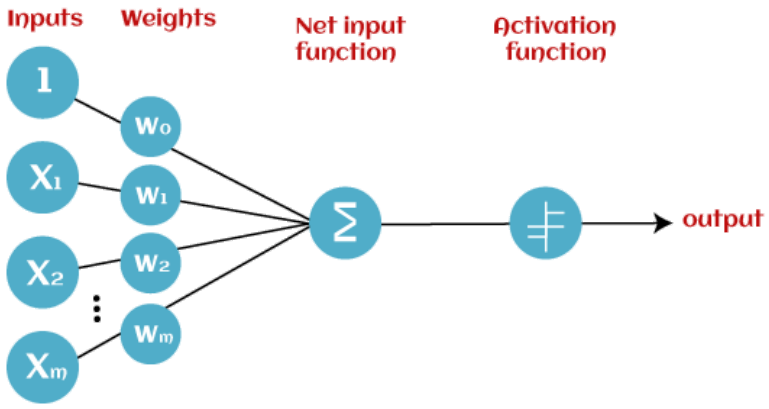
UNDERSTANDING THE PROBLEM STATEMENT

- Perceptron Classifier is an 2 class binary classification machine learning algorithm.
- It is simple type of neural network model.

> 💡 Activation = weights * input + Bias

- If the value of Activation is larger then 0.0 then it will `predict : 1`
- If the value of Activation is less then or equal to 0.0 then it will `predict: 0`
- The Basic components of the Perceptron Model are :
  - Input Layer.
    - Accepts initial data into the system for further processing.
  - Weights and Bias.
    - Weight is an parameter which represents the strength of the connection between the units.
    - Weight is directly proportional to the strength of the associated input neuron in deciding the output.
    - Bias can be considered as the line of intercept in an linear equation.
  - Net Sum.

- Activation Function.
  - It is the final and important component that help to determine whether the neuron will fire or not.
  - It is primary considered as the step function



WORKING OF PERCEPTRON MODEL

- In first step, it multiply all input values with corresponding weight values and then add them to determine the weighted sum.
- Mathematically , it will be represented as follows :

$$\sum wi * xi = x1 * w1 + x2 * w2 + \ldots wn * xn$$

- Now in that stage , we will add the special term bias ‘b’ in weighted sum in order to increase the model performance.

$$\sum wi * xi + b$$

- In the second stage , an activation function is applied with the above mentioned weighted sum , which gives us the output either in the binary form or an continuous value as follows :

$$Y = f(\sum wi * xi + b)$$

TYPES OF PERCEPTRON MODELS

- Single Layer Perceptron.
- Multi Layer Perceptron.

SINGLE LAYER PERCEPTRON

- It consists of feed-forward network and also contain threshold transfer function.
- The main objective of this model is to analyze the linearly separable objects with binary outcome.
- It do not contain recorded data, it begins with inconstantly allocated input for weights parameters.
- Further , it sums up all the values and after adding all the inputs if the total sum of all inputs is more then pre-determined then model gets activated and shows the output value of +1.

💡 Single Layer Perceptron can learn only linearly separable patterns

MULTILAYER PERCEPTRON

- It also has the same model structure but has a greater number of hidden layers.
- It is also known as Backpropagation Algorithm.
- It executes in 2 stages :
  - Forward Stage.
    - Activation function start from the input layer in the forward stage and terminate on the output layer
  - Backward Stage.
    - Weights and bias values are modified as per the model requirements.
    - In this stage , the error between the actual output and demanded originated backward on the output layer and ended in the input layer.

Advantages of Multi-Layer Perceptron:

- A multi-layered perceptron model can be used to solve complex non-linear problems.
- It works well with both small and large input data.
- It helps us to obtain quick predictions after the training.
- It helps to obtain the same accuracy ratio with large as well as small data.

Disadvantages of Multi-Layer Perceptron:

- In Multi-layer perceptron, computations are difficult and time-consuming.
- In multi-layer Perceptron, it is difficult to predict how much the dependent variable affects each independent variable.
- The model functioning depends on the quality of the training.

CODE OF THE FOLLOWING

- It is the initial step , in which all the dependencies are imported and also EDA is applied on the data.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

my_data = pd.read_csv('diabetes.csv')
x = my_data.iloc[:,[0,1,2,3,4,5,6,7]].values
y = my_data['Outcome']
y_train , y_test = train_test_split(x,y,test_size=0.25,random_state=63)
```

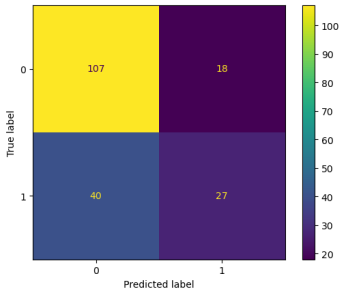
- Now in this step Perceptron classifier is setup.

```
Classifier = Perceptron(random_state=0)
Classifier.fit(X_train, y_train)
y_pred = Classifier.predict(X_test)
```

- Now in this step the accuracy score and confusion matrix is displayed.

```
In [13]: score = accuracy_score(y_test,y_pred)
In [14]: print(score)
         0.6979166666666666
In [15]: cm = confusion_matrix(y_test,y_pred)
In [16]: disp = ConfusionMatrixDisplay(confusion_matrix=cm)
In [17]: disp.plot()
```

```
score = accuracy_score(y_test,y_pred)
print(score)
cm = confusion_matrix(y_test,y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
```



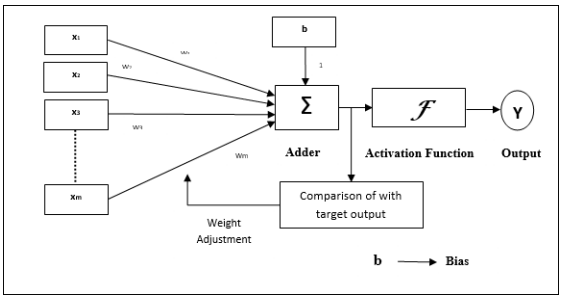
PROJECT 5

Create an ADALINE (Adaptive Learning) model for logical OR function

UNDERSTANDING THE PROBLEM STATEMENT

- Adaline is an network having an single linear unit , it was discovered in 1960.
- It uses bipolar activation function.
- It uses delta rule for training to minimize the Mean-Squared-error between the actual output and the desired/target output.
- The weights and bias are adjustable.

ARCHITECTURE



- The one extra thing which is present in this structure is an extra feedback loop with the help of which the actual output is compared with the desired/target output.
- After comparison on the basis of training algorithm , the weights and bias are updated.

TRAINING ALGORITHM

Step 1 − Initialize the following to start the training −

- Weights
- Bias
- Learning rate α

For easy calculation and simplicity, weights and bias must be set equal to 0 and the learning rate must be set equal to 1.

Step 2 − Continue step 3-8 when the stopping condition is not true.

Step 3 − Continue step 4-6 for every training vector **x**.

Step 4 − Activate each input unit as follows −

$$xi = si(i = 1ton)$$

Step 5
− Now obtain the net input with the following relation −

$$yin = b + \sum_{i}^{n} xi.wi$$

Here **'b'** is bias and **'n'** is the total number of input neurons.

Step 6 − Apply the following activation function to obtain the final output.

$$f(y_{in}) = \begin{cases} 1 & if\, y_{in} > \theta \\ 0 & if\, -\theta \leqslant y_{in} \leqslant \theta \\ -1 & if\, y_{in} < -\theta \end{cases}$$

Step 7 − Adjust the weight and bias as follows −

Case 1 − if **y ≠ t** then,

$$w_i(new) = w_i(old) + \alpha t x_i$$

$$b(new) = b(old) + \alpha t$$

Case 2− if **y = t** then,

$$wi(new) = wi(old)$$

$$b(new) = b(old)$$

Here **'y'** is the actual output and **'t'** is the desired/target output.

Step 8 − Test for the stopping condition, which would happen when there is no change in weight.

CODE FOR THE FOLLOWING IS GIVEN BELOW

```
def adaline(alp,w1,w2,b):
    x1 = [-1,-1,1,1]
    x2 = [-1,1,-1,1]
    t = [-1,1,1,1]
    E = []
    for j in range(1,3):
        for i in range(4):
            yin = x1[i]*w1 + x2[i]*w2 + b
            w1 = w1 + alp*(t[i]-yin)*x1[i]
            w2 = w2 + alp*(t[i]-yin)*x2[i]
            b = b + alp*(t[i]-yin)
            Error  = (t[i]-yin)**2
            E.append(Error)
        print("weights and b")
        print(w1,w2,b)
        print("Emean",j)
        print(sum(E)/4)
        E.clear()
```

- Now , we will write the calling code as follows :

```
adaline(0.1,0.1,0.1,0.1)
```

- The output of the following is given below:

```
weights and b
0.24011 0.22031000000000003 0.25811
Emean 1
0.7844263525
weights and b
0.32722905100000005 0.298519871 0.353982851
Emean 2
0.5325514092613155
```

DASHBOARD TOPICS

HEBBIAN LEARNING

- Easiest learning rule in the neural networks.
- It is used for the pattern classification.
- It is single layer neural network.
- It has one input and one output layer.
- Input layer can have many units (say n) but output layer has one unit only.
- Hebbian tule works by updating the weights between neurons in the neural network for each training sample.

Hebbian Learning Rule Algorithm :

1. Set all weights to zero, w= 0 for i=1 to n, and bias to zero.
2. For each input vector, S(input vector) : t(target output pair), repeat steps 3-5.
3. Set activations for input units with the input vector X = S for i = 1 to n.
4. Set the corresponding output value to the output neuron, i.e. y = t.
5. Update weight and bias by applying Hebb rule for all i = 1 to n:

$$w_i\,(new) = w_i\,(old) + x_i\,y$$

$$b\,(new) = b\,(old) + y$$

EPPOCH IN MACHINE LEARNING

An epoch is **when all the training data is used at once and is defined as the total number of iterations of all the training data in one cycle for training the machine learning model**

Another way to define an epoch is the number of passes a training dataset takes around an algorithm

DIFFERENCE BETWEEN VARIENCE AND BIAS

Signs of a High Bias ML Model	Signs of a High Variance ML Model
failure to capture data trends	noise in data set
underfitting	overfitting
overly simplified	complexity
high error rate	forcing data points together

DIFFERENCE BETWEEN OVERFITTING AND UNDERFITTING

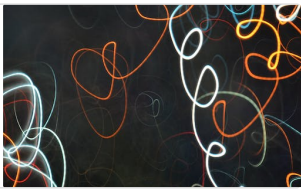
Techniques to fight underfitting and overfitting	
Underfitting	**Overfitting**
More complex model	More simple model
Less regularization	More regularization
Larger quantity of features	Smaller quantity of features
More data can't help	More data can help

VIVA QUESTIONS

35 Artificial Neural Network (ANN) Interview Questions (EXPLAINED) for Data Scientists | MLStack.Cafe

The human brain is composed of 86 billion nerve cells called neurons. The idea of ANNs is based on the belief that the working of the human brain can be imitated using silicon and wires as living neurons and dendrites.

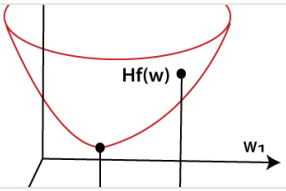
https://www.mlstack.cafe/blog/neural-network-interview-questions



Artificial Neural Network Interview Questions

Neural Network is a sequence of an algorithm that gives its best to recognize the underlying relationship in a data-set through a process. The artificial neural network has several differences from biological brains. These networks play a crucial role in deep learning. Biological Neural Network is

https://www.tutorialandexample.com/artificial-neural-network-interview-questions



REPOSITORY LINK

- All codes and dataset is mentioned in the repository bookmarked down below take a look on it.

https://github.com/Harshvardhan1609/Neural-Network-basic-codes