the main goals into sub-goals that are linked with actionable task.

* Business Transformation → this technique is used to implement transformat^n projects. If there are any desired changes in the current state of a busi. project, MEA is applied to establish the new prouses to be implemented. The processes are split into sub-processes to enhance effective implementation.

* Gap Analysis → is the comparison b/w the current performance & the reqd. performance. MEA is applied in this field to compare the existing technology & the desired tech. In org's. various op's are applied to fill the existing gap in technology.

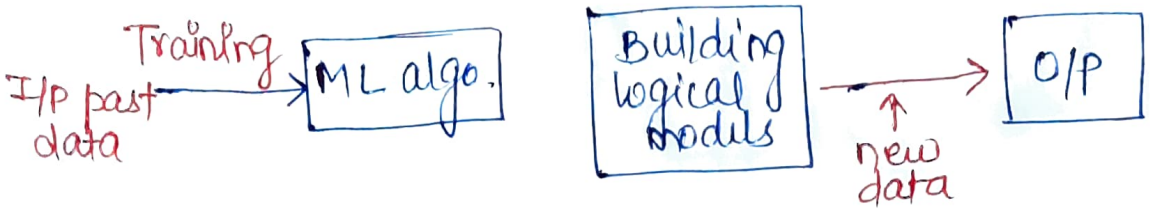
MACHINE LEARNING

is a branch of AI which enables m/cs to learn from past data or experiences without being explicitly programmed.

ML enables a comp. sys. to make predictions or take some decisions using historical data without being explicitly programmed.

ML process: ML sys. learns from historical data, builds the prediction moduls & whenever it receives

new data, predicts the o/p for it.



Goal of ML

* primary objective is to develop general purpose algo. in practical value.
* Main purpose is to study & design the algos that can be used to produce the predicates from the given dataset.

ML can also be used to:

→ enhance CX service
→ predict journey times
→ predict how long Jobs may take
→ Behaviour of Market

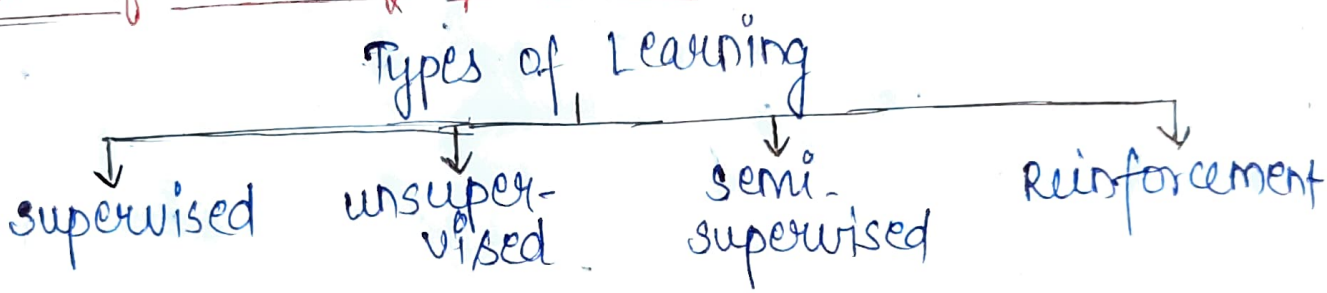
Applications of ML:

1. Image Recognition
2. speech "
3. Traffic prediction
4. product Recommendatn
5. Self-driving cars
6. email spam & Malware filtering
7. virtual personal assistant
8. Online fraud Detection
9. stock market trading
10. Medical Diagnosis.

Challenges for ML:

1. Data collection
2. Less Amount of Training Data
3. Non-representative " "
4. poor Quality of data
5. Irrelevant / unwanted features.

Classification of ML



(1) Supervised Learning

Its a type of ML method in which we provide sample labeled data to the ML sys. in order to train it, & on that basis, it predicts the o/p.

(2) Unsupervised Learning

is a learning method in which a M/c learns without any supervision.

(3) semi-supervised learning

its working lies b/w supervised & unsupervised techniques. we use these, when we are dealing with a data which is a little bit labeled & rest large portion of it is unlabelled. Mostly applicable

in case of image data-sets where usually all images are not labeled.

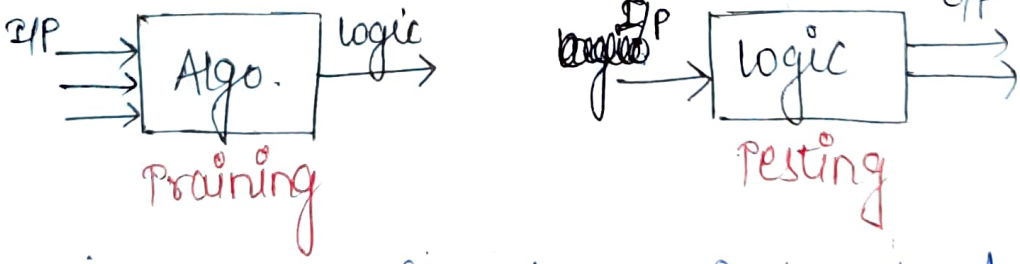
(4) Reinforcement Learning

It's a feedback based " method in which a learning agent gets a reward for each right action & gets penalty for each wrong action.

SUPERVISED LEARNING

* a m/c is trained using labeled data. Datasets are said to be labeled when they contain both I/p and o/p parameters, and on that basis, it predicts the o/p.
This implies that some data is already tagged with the correct answer.

*



ex Basket filled with diff. kinds of fruits.

Classification : Supervised learning

1. Classification → A classificat" prob. is when the o/p variable is a category such as 'Red', 'Blue', 'disease' and 'no disease'.
2. Regression → when the o/p variable is a real value, such as 'dollars', 'weight'.

Categories of supervised learning:

* Regression
* classification
* K-NN (nearest neighbors)
* Support Vector M/c (SVM)
* logistic Regression
* Naive Bayes classifiers
* Decision Tree

UNSUPERVISED LEARNING

* m/c learns without any supervision
* training is provided to the m/c with the set of data that has not been labeled, classified.
* No training will be given to the m/c. ∴ the m/c is restricted to find the hidden struc. in unlabeled data by itself.
* classification of unsupervised learning:

1) Clustering →
A clustering prob. is where we want to discover the inherent groupings in the data, such as grouping cxes by purchasing behavior.

2) Association →
Discover rules that describe large portions of data such as ppl that buy X also tend to buy Y.

example: suppose it's given an image having both dogs & cats which it has never seen.

m/c has no idea about the features of dogs & cats. So, we can't categorize, but it can categorize them acc. to their similarities, patterns & differences.

categorize the above image into two parts; first may contain all pics having dogs & second may contain all pics having cats.

Clustering:

* find patterns in the data that we are working on.
* it may be the shape, size, color etc. which can be used to group data items or create clusters.

popular algos of clustering are:

1) Hierarchical clustering - this algo builds clusters based on the similarity btw diff data points in the dataset.

2) K means clustering - algo. creates clusters of diff. data pts which are as homogeneous as possible by calculating the centroid of the cluster & making sure that the dist. btw this centroid & each data pt is as less as possible.

3) K-NN clustering also k/a lazy learner bcoz it learns only when algo. is given a new data pt. It works well with smaller datasets as large datasets take time to learn.

Association :

In this type of learning, we find the dependencies of one data item to another data item & map them such that they help you profit better.

1) Apriori Algo. →

* based on breadth - first search
* this algo. maps the dependency of one data item with another which can help us understand what data item influences the possibility of something happening to the other data item.
* ex: bread influences the buyer to buy milk & eggs. so that mapping helps ↑es profit for the store.

2) FP - Growth Algo. → (frequent pattern)

* FP algo. finds the count of the pattern that has been repeated, adds that to a table & then finds the most possible item & sets that as the root of the tree.
* this algo is faster than Apriori as the support is calculated & checked for ↑ing iterations rather than creating a rule & checking the support from the dataset.

$$\text{support} = \frac{\text{how many times the items are brought together}}{\text{total no. of Tx's}} * 100\%$$

$$\text{Confidence}_{(X \to Y)} = \frac{\text{how many times items X, Y are brought together}}{\text{how many times an item X is brought}} * 100\%$$

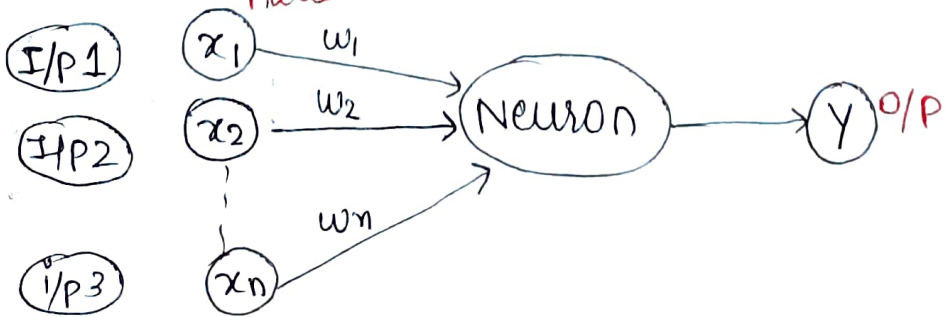
ARTIFICIAL NEURAL NETWORK

ANN is derived from Biological NN that develop the struc. of a human brain. The Human brain has hundreds of billions of cells k/a Neurons.

Each Neuron is made up of a cell body that is responsible for processing info. by carrying info. towards (i/ps) and away (o/ps) from the brain.

What is ANN ?

- ANN are a special type of ML algo. that are modeled after the human brain.
- How the neurons in our nervous sys. are able to learn from the past data & provide responses & similarly, ANN is able to learn from the data & provide responses in the form of predictions or classifications.
- ANNs are Non-linear statistical models which displays a complex rel'ship b/w i/ps & o/ps to discover a new pattern.
- adv. - it learns from the example data sets.
- ANN is also capable of taking sample data rather than the entire dataset to provide the o/p result.



ANN ARCHITECTURE

The f^n ing of the ANN is similar to the way neurons work in our nervous sys.

In NN, there are 3 essential layers :

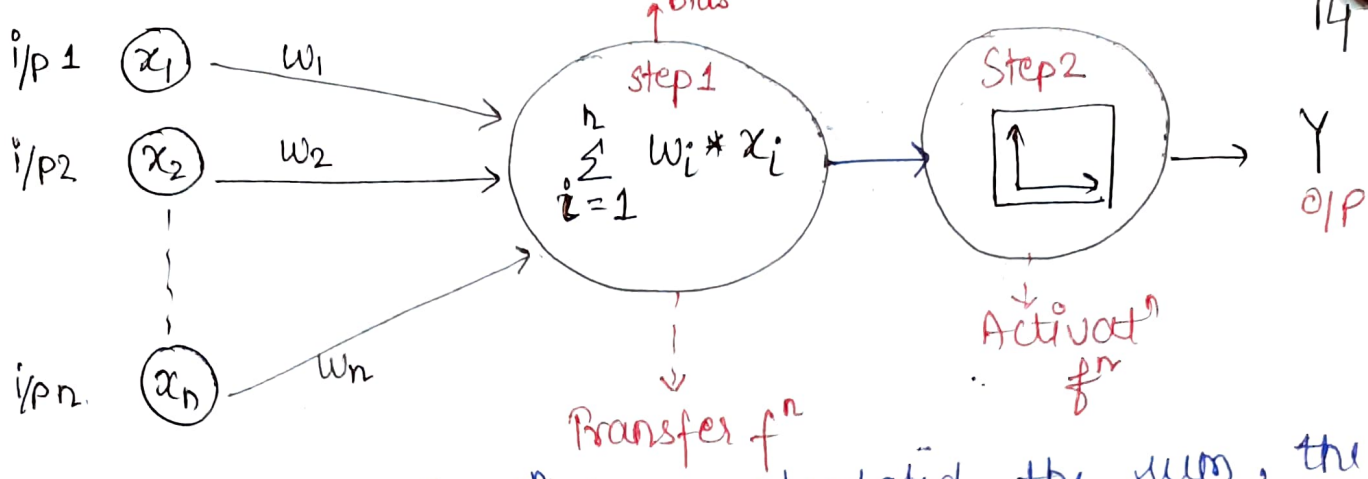
1) INPUT LAYER → first layer of an ANN that receives the i/p info. in the form of various texts, nos., audio files, image etc.

2) HIDDEN LAYER → In the middle of ANN model are the Hidden Layers. There can be a single hidden layer, as in the case of a perceptron or multiple hidden layers. These layers perform various types of mathematical computat^n on the i/p data & recog-nize the patterns that are part of.

3) OUTPUT LAYER → we obtain the result that we obtain through rigorous computations performed by the middle layer.

In a NN, there are multiple parameters and hyperparameters that affect the performance of the model. The o/p of ANNs is mostly dependent on these parameters. Some of them are : weights, biases, learning rate, batch size etc.

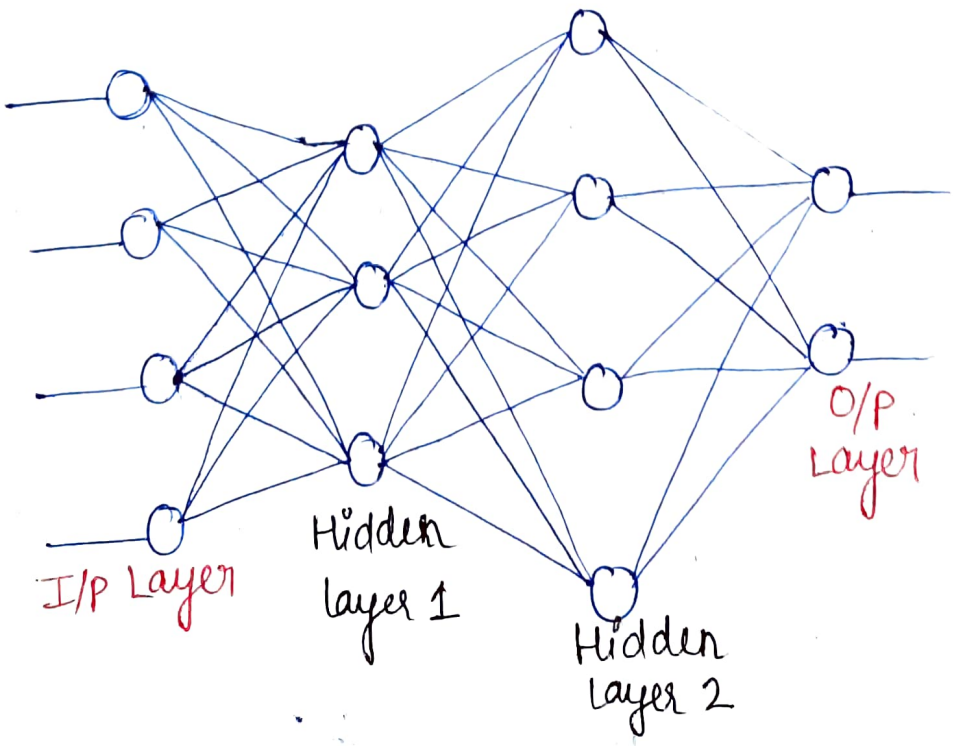
Each node in the n/w has some weights assigned to it. A Transfer f^n is used for calculating the weighted sum of the i/ps & the bias.



Aftr the transfer f^n has calculated the sum, the Activation f^n obtains the result. Based on the o/p received, the activat^n f^n fire the appropriate result from the node.

ex: if o/p received is above 0.5, the activat^n f^n fires a 1 otherwise 0.

Based on the value that the node has fired, we obtain the final o/p. Then, using the error f^n(s), we calculate the discrepancies b/w predicted o/p and resulting o/p & adjust the wt. of the NN through a process known as Backpropagation.



Neural N/ws / ANN / simulated NN (SNN)

ANN are comprised of a node layers, containing an I/p Layer, one or more hidden layers & an O/p Layer. Each node connects to another & has an associated wt. and threshold. If the O/p of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the n/w. Otherwise, No data is passed along to the next layer of the n/w.

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

$$\text{Output} = f(x) = \begin{cases} 1 & \text{if } \sum w_i x_i + b > 0 \\ 0 & \sum w_i x_i + b < 0 \end{cases}$$

This process of passing data from one layer to the next layer defines this NN as a Feedforward N/w.

ex. whether you should go surfing or not. Assume that 3 factors influencing your decision making:

1. Are the waves good ? (Yes : 1, No : 0)
2. Is the line-up empty ? (Yes : 1, NO : 0)
3. Has there been a recent shark attack ? (Yes : 0, No : 1)

Let's assume the follo. i/ps :

$x_1 = 1$, since the waves are pumping

$x_2 = 0$, since the crowds are out

$x_3 = 1$, " there hasn't been a recent shark attack

Now, assign some wts to determine importance

Large wt. - greater importance to decision or outcome

$w_1 = 5$, since large swells don't come around often

$w_2 = 2$, " you're used to the crowds

$w_3 = 4$, " you have a fear of sharks

$$\text{Y-hat} = (1*5) + (0*2) + (1*4) - 3 = 6$$

O/p would be 1, since 6 is greater than 0.

BACK PROPAGATION

Method of fine-tuning the wts of a NN based on the error rate obtained in the previous iteration. proper tuning of the wts allows you to reduce error rates & make the model reliable by ↑ing its generalization.

It is a short form for Backward propagation of errors. It's a std method of training ANN

It helps to calculate the gradient of a loss f^n wrt all wts in the n/w.

How Backpropagation Algo works?

It computes the gradient of the loss f^n for a single wt. by the chain rule.

It efficiently computes one layer at a time,

How Backpropagation algo. works:

1. I/Ps X, arrive through the preconnected path
2. i/p is modeled using real weights W. The weights are usually randomly selected.
3. Calculate the o/p for every neuron from the I/p layer, to the hidden Layers, to the o/p layer.
4. Calculate the error in the o/ps:

$$\text{Error} = \text{Actual o/p} - \text{Desired o/p}$$

5. Travel back from the o/p layer to the hidden layer to adjust the wt.(s) such that the error is ↓ed.

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

Why we need Backpropagation ?

* It's fast, simple & easy to program
* It has No parameters to tune apart from the nos. of i/p
* It's a flexible method as it doesn't require prior knowledge about the n/w.
* Std method that generally works well.
* It doesn't need any spcl mention of the features of the f^n to be learned.

FEED FORWARD NETWORK

It's an Artificial NN where the nodes never form a cycle. This kind of NN has an i/p Layer, & an

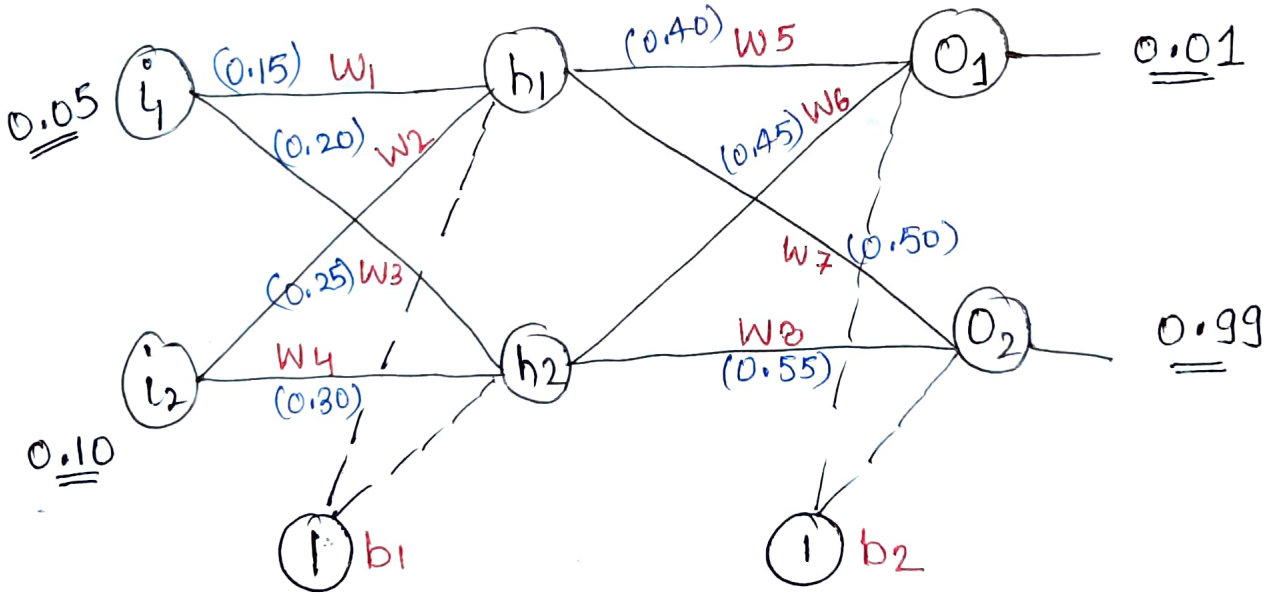
O/p layer.

Types of Backpropagation Network :

* **Static Backpropagation** → kind of BP n/w which produces a mapping of a static i/p for static o/p. Its useful to solve static classification issues like OCR. (optical character Recognition).

* **Recurrent Backpropagation** → Recurrent BP in data mining is fed forward until a fixed value is achieved. After that, the error is computed & propagated backward.

Example :-



Goal of backpropagation is to optimize the wt.cs so that the NN can learn how to correctly map arbitrary i/ps to o/ps.

Given: I/Ps ⇒ 0.05 & 0.10

O/Ps ⇒ 0.01 & 0.99

Calculate total net o/p for h_1 :

$net_{h_1} = w_1 i_1 + w_2 i_2 + b_1 * 1$

$net_{h_2} = 0.3925$

$= 0.15 * 0.05 + 0.20 * 0.10 + 0.35 * 1 = 0.3775$

then squash it using the logistic f^n to get th o/p of h_1 :

$$out_{h_1} = \frac{1}{1 + e^{-net_{h_1}}} = \frac{1}{1 + e^{-0.3775}} = 0.593269$$

$Out_{h_2} = 0.596884$

Repeat this process for the o/p Layer neurons, using the o/p from the hidden layer neurons as i/ps :

$net_{o_1} = w_5 * out_{h_1} + w_6 * out_{h_2} + b_2 * 1$

$net_{o_1} = 0.4 * 0.5932699 + 0.45 * 0.596884 + 0.6 * 1$

$net_{o_1} = 1.105905$

$net_{o_2} = 1.22492091$

$$Out_{o_1} = \frac{1}{1 + e^{-net_{o_1}}} = 0.75136507$$

$Out_{o_2} = 0.772928465$

$$\text{Total Error} = \sum \frac{1}{2}(\text{target} - \text{o/p})^2$$

$E_{o_1} = \frac{1}{2}(target_{o_1} - out_{o_1})^2 = \frac{1}{2}(0.01 - 0.751365)^2 = 0.274811083$

$E_{o_2} = 0.023560026$

$E_{total} = E_{o_1} + E_{o_2} \Rightarrow 0.274811 + 0.023560 = 0.298371109$

SUPPORT VECTOR MACHINE

SVM is a simple & powerful supervised ML algo. that can be used for building both Regression & classification models. SVM algo. can perform really well with both linearly & non-linearly separable datasets.

Types:

1. **Linear or Simple SVM** → is used for linearly separable data. If dataset can be classified into two classes with a single straight line, then data is considered to be linearly separable data & the classifier is referred to as the linear SVM. Typically, used for linear Regression & classification problems.

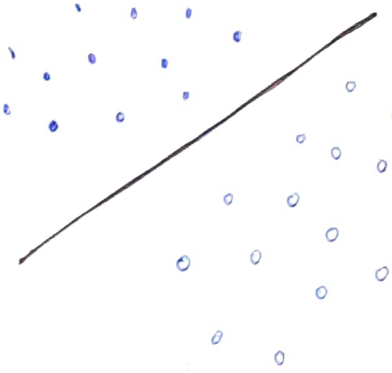
2. **Non-Linear or Kernel SVM** → used for Non-linear separated data i.e. dataset that can't be classified by using a straight line. The classifier used in this case is referred as Non-linear SVM classifier. It has more flexibility for non-linear data bcoz more features can be added to fit a hyperplane instead of a 2-D space.

Example: (Linear SVM)

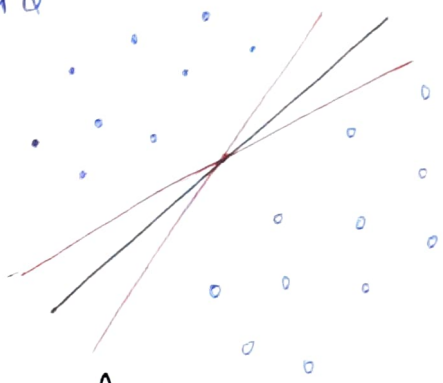
SVM algo. is based on the concept of 'Decision planes' where hyperplanes are used to classify a set of given objects.

In given fig., have two sets of data. These datasets can be

separated easily with the help of a line, kya Decision Boundary.

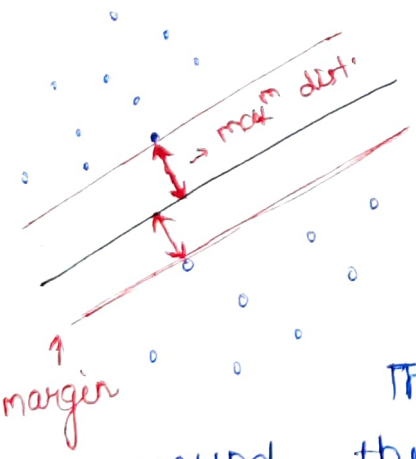


But there can be several decision boundaries that can divide the data points without any errors.



All decision boundaries classify the datasets correctly. But how do we pick the best decision boundary?

The best decision boundary is the one that has a max^m distance from nearest pnts of these 2 classes.



Also, remember that the nearest pnts from the optimal decision boundary that maximize the dist. are kya Support Vectors.

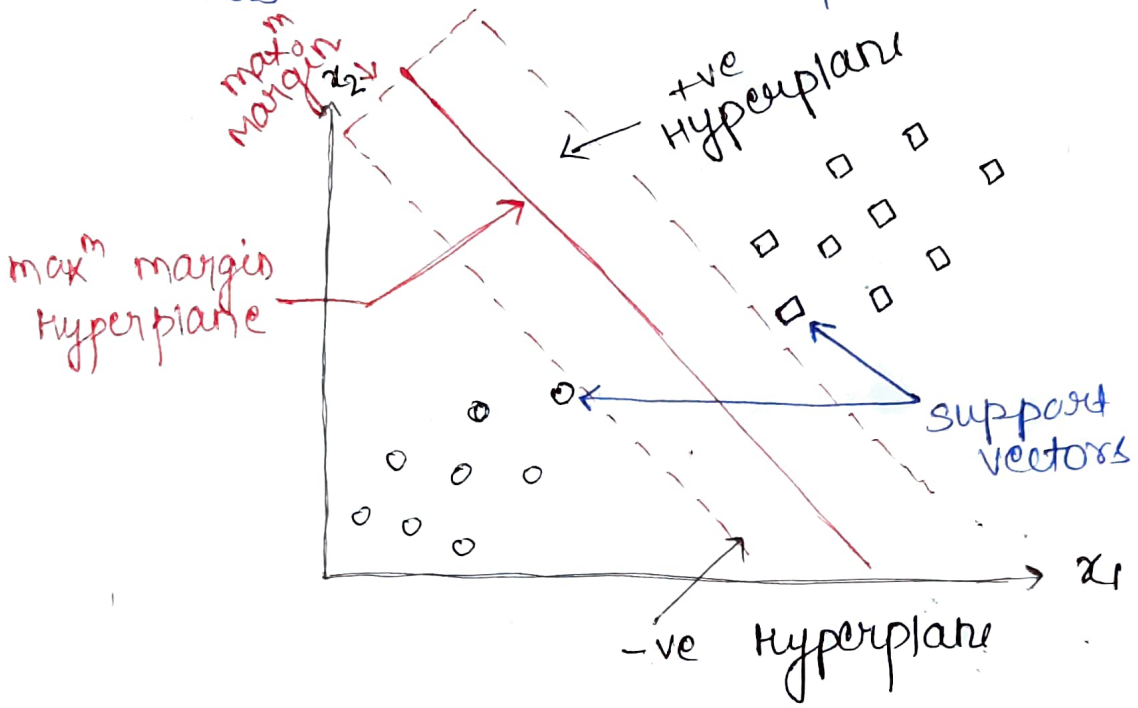
The region that the closest pnts dying around the decision boundary is known as Margin. That's why the decision boundary of a SVM model is known as the max^m margin classifier or the max^m margin Hyperplane.

How a SVM algo. model works:

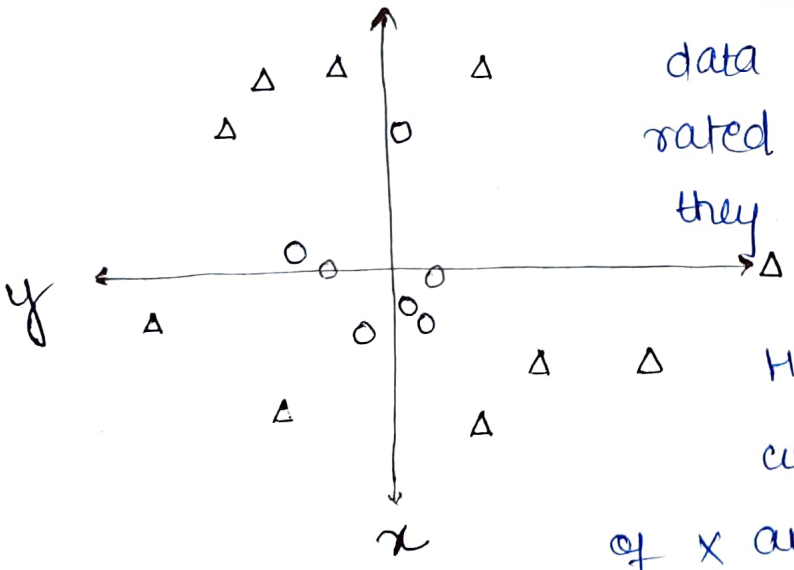
* First, it finds lines or boundaries that correctly

classify the training dataset.

* then, from those lines or boundaries, it picks one that has the max^m dist. from the closest data pnts.

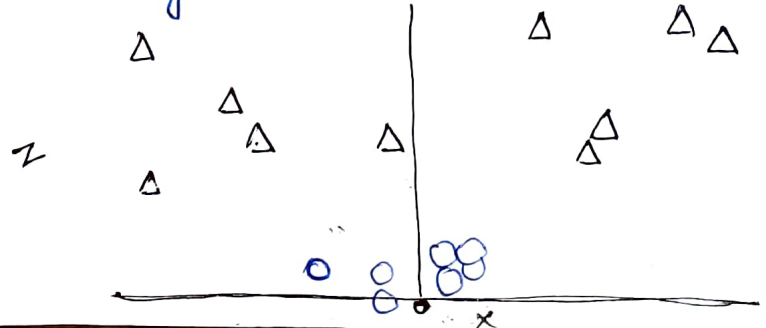


Example: (Non-linear SVM)

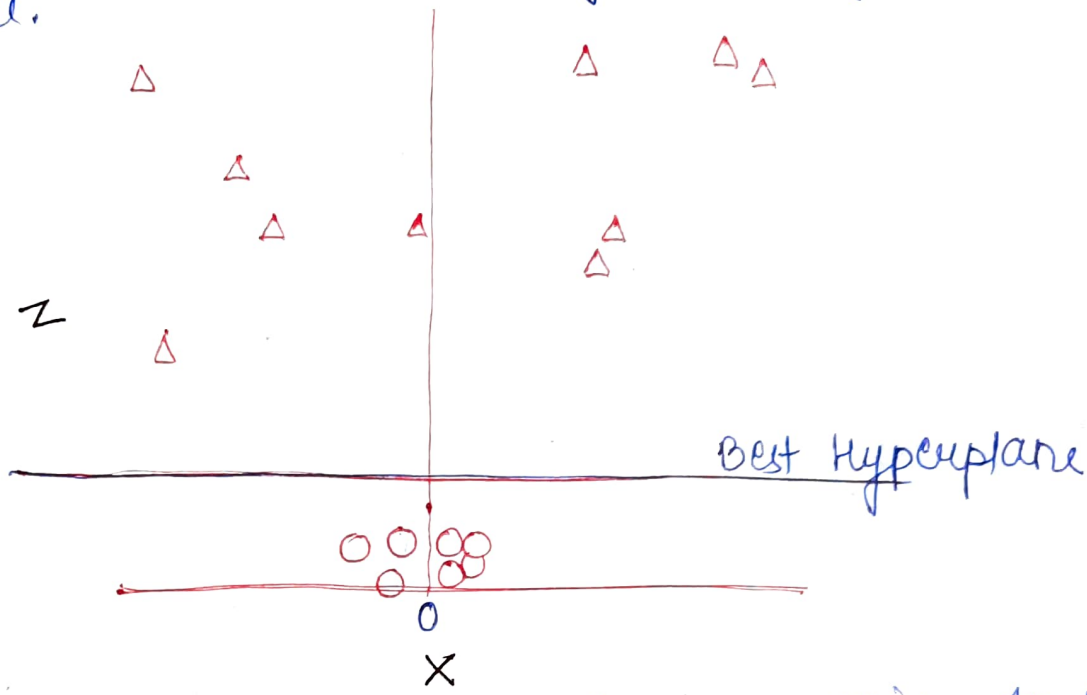


These are two classes of data pnts which can't be separated by a straight line. But they can be separated by a circular Hyperplane, Hence, we can introduce a co-ordinate z with the help of x and y where $z = x^2 + y^2$

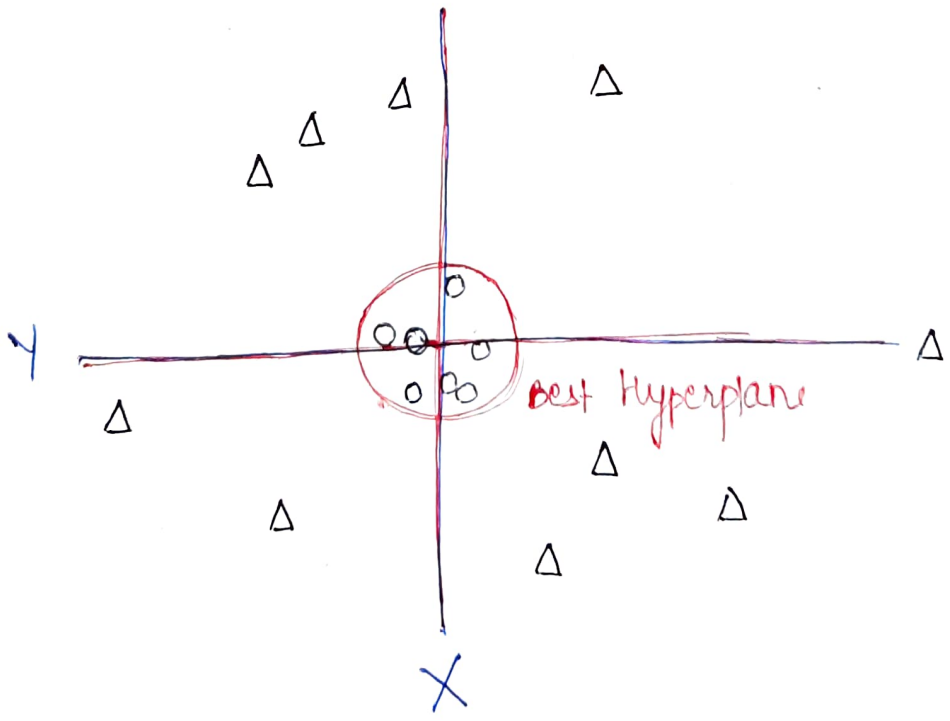
Now, after introducing 3rd dimension, the graph changes to:



Now, the above depiction of data pnts is linearly separable & can be separated by a straight-line Hyperplane.



This representat" is in 3-D with a z-axis. In 2-D the graph looks like:



This is what, a non-linear SVM does! It hypothetically takes the data pnts to a higher dimension, so that they are linearly separable in that dimension & then the algo. classifies them!

Advtgs. of SVM: * has high level of accuracy
* works well with limited datasets
* Kernel SVM contains a non-linear transformat° f° to convert the complicated non-linearly separable data into linearly separable data.
* effective on datasets that have multiple features
* effective when no. of features are greater than no. of data points.

Disadvtgs:
* doesn't work well with larger datasets
* sometimes, training time can be high
* if no. of features is significantly greater than no. of data pnts, it's crucial to avoid overfitting when choosing kernel f°cs & regularizat° terms
* Best works on small sample sets due to its high training time.

REINFORCEMENT LEARNING

Terminology used in RL:

* Agent → it's an assumed entity which performs action in an environ. to gain some reward.
* Environment (e) → a scenario that an agent has to face
* Reward (R) → an immediate return given to an agent when he/she performs specific action or task.
* state (s) → refers to the current situation returned by the environ.

* policy (π) → it's a strategy which applies by the agent to decide the next action based on current state.
* value (V) → it's expected long term return with discount, as compared to the short term reward.
* value function → it specifies the value of a state that is the total amt of reward. It's an agent which should be expected beginning from that state.
* Model of the environ. → this mimics the behavior of the environ. It helps you to make inferences to be made & also determine how the environ. will behave.
* Model based methods → method for solving RL probs. which use model based methods.
* Q-value or Action value (Q) → value is quite similar to value. The only diff b/w two is that it takes an add'al parameter as a current action.

Characteristics of RL

* No supervisor, only a real no. or reward signal
* sequential decision making
* FB is always delayed, not instantaneous
* Time plays a crucial role in RL
* Agent's actions determine the subsequent data it receives

Types:- * +ve RL
* -ve RL

Positive: defined as an event, that occurs bcoz of specific behavior. It ↑es the strength & the frequency of the behavior & impacts +vely on the action taken by the agent.
This type of Reinforcement helps you to maximize performance & sustain change for a more extended period.

Negative: defined as strengthing of behavior that occurs bcoz of -ve cond^n which should have stopped or avoided. It helps you to define the min^m stand of performance.

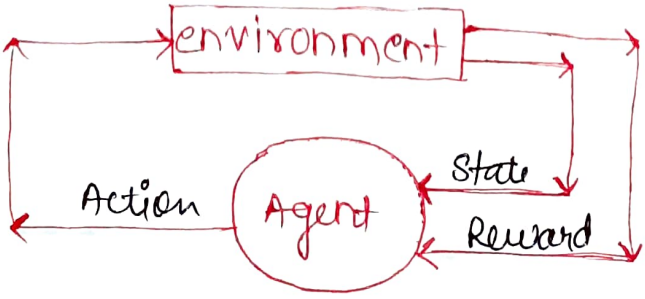
Learning Modls of Reinforcement

1) Markov Decision process 2) Q-Learning

Markov Decision process: following parameters are used to get a solution:

* set of actions (A) * set of states (S)
* Reward (R) * policy (π) * Value (V)

The mathematical approach for mapping a sol^n in Reinforcement learning is recon as a Markov Decision process (MDP)



Q- Learning: is a value based method of supplying info. which action an agent should take.

Why use RL?

* it helps you to find which situation needs an action
* helps you to discover which action yields the highest reward over the longest period.
* also provides the learning agent with a reward f
* also allows it to figure out the best method for obtaining large rewards.

Challenges:

* parameters may affect the speed of learning
* feature / reward design which should be very involved.
* Realistic environ. can have partial observability
* Too much reinforcement may lead to an overload of states which can diminish the results.
* Realistic environ. can be non-stationary.

ADAPTIVE LEARNING

Adaptive ML builds on traditional ML to create a more advanced sol^{n} to real time environ(s). with variable data.

AML can adapt to rapidly changing data sets, making it more applicable to real-world situations.

Adaptive ML is more robust & efficient than tra- ditional ML & incorporates agility, ↑ed accuracy & greater sustainability.

AML can process large quantities of data while its op'al cond'cs) can be more easily adjusted as the need of the company using it changes.

It can quickly adapt to new info. & provide real time insight into how that data can be used.

It uses single channel structure, which means it collects new info. in real time. Its also able to use diff. techniques for gathering data as well as diff. ways of grouping & analyzing that data.

It can change acc. to new data & provide rapid insights.

Benefits of Adaptive ML:

1. More efficient → use to find relevant sol'cs) faster. Since, it works on a single channel, it can creat faster sol'cs with newest data at hand.
2. More pertinent data → Rather than using old, static data, Adaptive ML is constantly taking in new, more relevant data. It can change its behavior based on this new data.
3. Able to learn from the past → the longer an Adaptive ML prog. runs, the more it learns. It can lower the chances of repeating a mistake by