ARTIFICIAL INTELLIGENCE & MACHINE LEARNING FUNDAMENTALS

UNIT I INTELLIGENT AGENT AND UNINFORMED SEARCH

Introduction - Foundations of AI - History of AI - The state of the art - Risks and Benefits of AI -Intelligent Agents - Nature of Environment - Structure of Agent - Problem Solving Agents -Formulating Problems - Uninformed Search - Breadth First Search - Dijkstra's algorithm or uniform-cost search - Depth First Search - Depth Limited Search

PART-A

1.What is Artificial Intelligence?

Artificial Intelligence is the study of how to make computers do things which at the moment people do better.

2. What is an agent?

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

3.What are the various agent programs in intelligent systems?

1)Simple reflex agents

2)Model-based reflex agents

3)Goal-based agents

4)Utility-based agents

4. List some of the uninformed search techniques.

The uninformed search strategies are those that do not take into account the location of the goal. That is these algorithms ignore where they are going until they find a goal and report success. The various uninformed search strategies are

- Breadth-first search
- Uniform-cost search
- Depth-first search
- Depth-limited search
- Iterative deepening depth-first search
- Bidirectional search

5. What is CSP? CSP are problems whose state and goal test conform to a standard structure and very simple representation. CSPs are defined using set of variables and a set of constraints on those variables. The variables have some allowed values from specified domain. For example – Graph coloring problem.

6. What are the advantages of heuristic function?

Heuristics function ranks alternative paths in various search algorithms, at each branching step, based on the available information, so that a better path is chosen. The main advantage

of heuristic functions is that it guides for which state to explore now, while searching. It makes use of problem specific knowledge like constraints to check the goodness of a state to be explained. This drastically reduces the required searching time.

7) Evaluate performance of problem-solving method based on depth-first search algorithm?

DFS algorithm performance measurement is done with four ways–

1) Completeness–It is complete(guarantees solution)

2) Optimality– it is not optimal.

3) Time complexity–It's time complexity is O(b).

4) Space complexity–its space complexity is O(bd+1)

8. Differentiate toy problems and real world problems?

A toy problem is intended to illustrate various problem solving methods. It can be easily used by different researchers to compare the performance of algorithms. A real world problem is one whose solutions people actually care about.

9. What are the components of well-defined problems? (or)

What are the four components to define a problem? Define them?

The four components to define a problem are,

1) Initial state–it is the state in which agent starts in.

2) A description of possible actions – it is the description of possible actions which are available to the agent.

3) The goal test –it is the test that determines whether a given state is goal (final) state.

4) A path cost function– it is the function that assigns a numeric cost(value) to each path. The problem-solving agent is expected to choose a cost function that reflects its own performance measure.

10. What are various applications of AI? Or What can AI do today?

- ✔ Robotic vehicles
- ✔ Speech recognition
- ✔ Autonomous planning and scheduling
- ✔ Game playing
- ✔ Spam fighting
- ✔ Logistics planning
- ✔ Robotics
- ✔ Machine Translation

PART –B

1.Explain Depth First Search Algorithms?

DFS is one of the recursive algorithms we know. It traverses the graph or a tree depth-wise. Thus it is known to be a depth-first search algorithm as it derives its name from the way it functions. The DFS uses the stack for its implementation. The process of search is

similar to BFS. The only difference lies in the expansion of nodes which is depth-wise in this case.Unlike the BFS, the DFS requires very less space in the memory because of the way it stores the nodes stack only on the path it explores depth-wise.

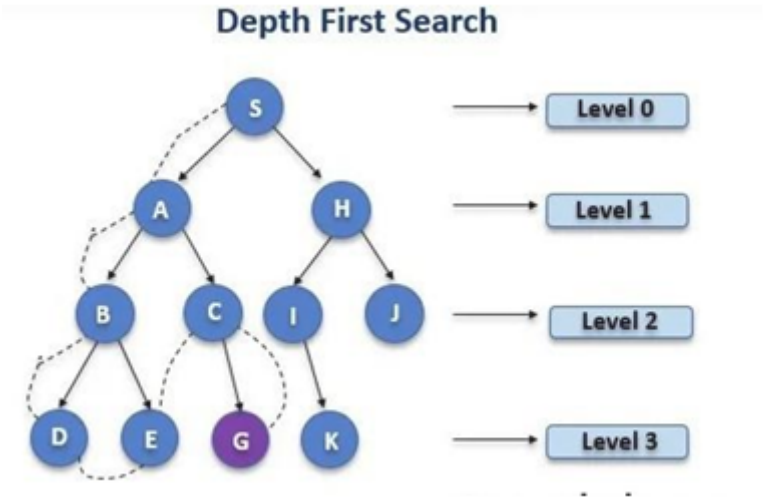
• In comparison to BFS, the execution time is also less if the expansion of nodes is correct. If the path is not correct, then the recursion continues, and there is no guarantee that one may find the solution. This may result in an infinite loop formation.

• The DFS is complete only with finite state space.

• Time Complexity is expressed as T(n) = 1+ n2+ n3+……+ nm=O(nm).

• The Space Complexity is expressed as O (bm).

• The DFS search algorithm is not optimal, and it may generate large steps and possibly high cost to find the solution.



2.Solve the Water Jug problem: you are given 2 jugs, a 4-gallon one and 3-gallon one. Neither has any measuring maker on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 gallons of water into 4-gallon jug? Explicit assumptions: A jug can be filled from the pump, water can be poured out of a jug onto the ground, water can be poured from one jug to another and that there are no other measuring devices available?

Given, two jugs, a -gallon one and a -gallon one.We have to give the full description. Let, A can contain 4 gallons & B can contain gallons of water First, fill B then pass the water to A Now A can contain one more gallon of water, Again fill container B fully. Completely fill A by transferring the water from B, B has gallons of water left.

The final answer is numerous source codes have been devised for solving Water Jug problems using recursion, searching and sorting algorithms. Here the initial state is (0, 0). The goal state is (2, n) for any value of n. State Space Representation: we will represent a state of the problem as a tuple (x, y) where x represents the amount of water in the 4-gallon jug and y represents the amount of water in the 3-gallon jug. Note that $0 \leq x \leq 4$, and $0 \leq y \leq 3$.

To solve this we have to make some assumptions not mentioned in the problem. They are:

- We can fill a jug from the pump.
- We can pour water out of a jug to the ground.
- We can pour water from one jug to another.
- There is no measuring device available.

Operators — we must define a set of operators that will take us from one state to another.

Sr.	Current State	Next State	Descriptions
1	(x, y) if x < 4	(4, y)	Fill the 4 gallon jug
2	(x, y) if y < 3	(x, 3)	Fill the 3 gallon jug
3	(x, y) if x > 0	(x - d, y)	Pour some water out of the 4 gallon jug
4	(x, y) if y > 0	(x, y - d)	Pour some water out of the 3 gallon jug
5	(x, y) if y > 0	(0, y)	Empty the 4 gallon jug
6	(x, y) if y > 0	(x, 0)	Empty the 3 gallon jug on the ground
7	(x, y) if x + y >= 4 and y > 0	(4, y - (4 - x))	Pour water from the 3 gallon jug into the 4 gallon jug until the 4 gallon jug is full
8	(x, y) if x + y >= 3 and x > 0	(x - (3 - y), 3)	Pour water from the 4 gallon jug into the 3-gallon jug until the 3 gallon jug is full
9	(x, y) if x + y <= 4 and y > 0	(x + y, 0)	Pour all the water from the 3 gallon jug into the 4 gallon jug
10	(x, y) if x + y <=3 and x > 0	(0, x + y)	Pour all the water from the 4 gallon jug into the 3 gallon jug
11	(0, 2)	(2, 0)	Pour the 2 gallons from 3 gallon jug into the 4 gallon jug
12	(2, y)	(0, y)	Empty the 2 gallons in the 4 gallon jug on the ground

There are several sequences of operations that will solve the problem. One of the possible solutions is given as:

Gallons in the 4-gallon jug	Gallons in the 3-gallon jug	Rule applied
0	0	2
0	3	9
3	0	2
3	3	7
4	2	5 or 12
0	2	9 or 11
2	0	--

3.Discuss in detail the uninformed search strategies and compare the analysis of various searches?

i).Breadth First Search

The procedure: In BFS root node is expanded first, then all the successor of root node are expanded and then their successor and so on. That is the nodes are expanded level wise

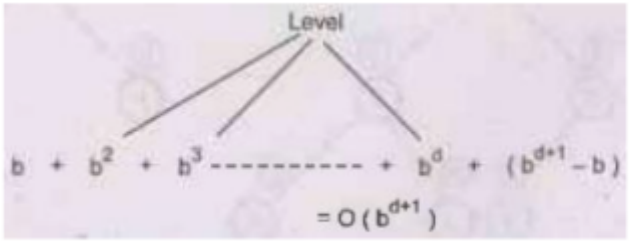
starting at root level.The implementation: BFS can be implemented using first, in first out queue data structure where fringe will be stored and processed. As soon as node is visited it is added to queue. All newly generated nodes are added to the end of the queue, which means that shallow nodes are expanded before deeper nodes.

The performance evaluation

1) Completeness: BFS is complete because if the shallowest goal node is at some finite depth d, BFS will eventually find it and will generate solution. (assuming that branching factor b is finite).

2) Optimality: The shallowest goal node is not necessarily optimal. BFS will yield optimal solution only when all actions have the same cost, let it be at any depth.

3) Time and space complexity: As the level of search tree grows more time is incurred. In general if search tree is at level d then O (b d+1) time is required, where b is the number of nodes generated from each node, starting at root node i.e. root generates b nodes and each b node generates b more and so on shown below.



Every node generated should remain in memory till its exploration. If branching factor 'b' is more then more memory will be required.

For example-

If branching factor b = 10 at some level d = 6. If we assume 10,000 nodes will be generated per second and per node 1000 bytes are required for storage. Then total generated nodes = 10^7 which will take 19 minutes but it will take 10 Gigabytes for storing them.We can conclude that for BFS space complexity is major issue concerned than time complexity. Time complexity is critical issue when depth of tree increases.revious percepts. Infinite loops might be avoided if the agent can randomize its actions, introducing some variability in its behavior.

4.Explain the Types of Agents?

1. Simple Reflex Agents

Simple reflex agents act solely based on the current percept and, percept history (record of past perceptions) is ignored by these agents. Agent function is defined by condition-action rules.

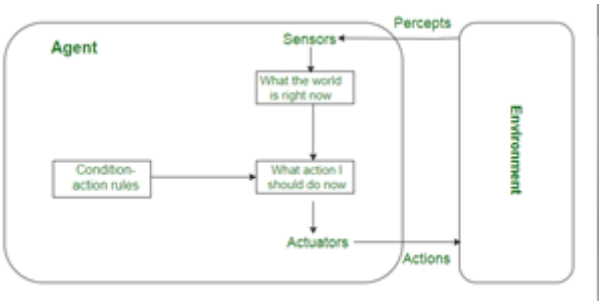
A condition-action rule maps a state (condition) to an action.

If the condition is true, the associated action is performed.

If the condition is false, no action is taken.

Simple reflex agents are effective in environments that are fully observable (where the current percept gives all needed information about the environment). In partially observable

environments, simple reflex agents may encounter infinite loops because they do not consider the history of p

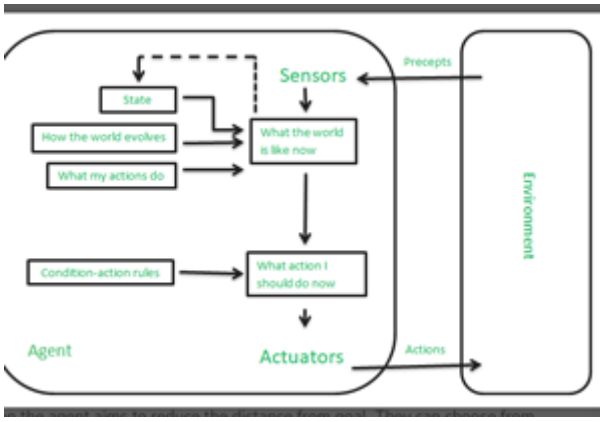


2. Model-Based Reflex Agents

Model-based reflex agents finds a rule whose condition matches the current situation or percept. It uses a model of the world to handle situations where the environment is only partially observable. The agent tracks its internal state, which is adjusted based on each new percept. The internal state depends on the percept history (the history of what the agent has perceived so far). The agent stores the current state internally, maintaining a structure that represents the parts of the world that cannot be directly seen or perceived.

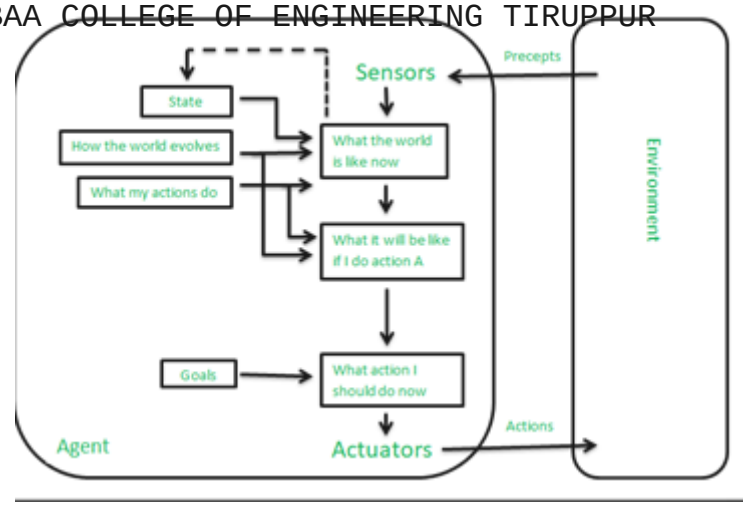
The process of updating the agent's state requires information about:

How the world evolves independently from the agent? and How the agent's actions affect the world?



3. Goal-Based Agents

Goal-based agents make decisions based on their current distance from the goal and every action the agent aims to reduce the distance from goal. They can choose from multiple possibilities, selecting the one that best leads to the goal state. Knowledge that supports the agent's decisions is represented explicitly, meaning it's clear and structured. It can also be modified, allowing for adaptability. The ability to modify the knowledge makes these agents more flexible in different environments or situations. Goal-based agents typically require search and planning to determine the best course of action.



4. Utility-Based Agents

Utility-based agents are designed to make decisions that optimize their performance by evaluating the preferences (or utilities) for each possible state. These agents assess multiple alternatives and choose the one that maximizes their utility, which is a measure of how desirable or "happy" a state is for the agent. Achieving the goal is not always sufficient; for example, the agent might prefer a quicker, safer, or cheaper way to reach a destination. The utility function is essential for capturing this concept, mapping each state to a real number that reflects the agent's happiness or satisfaction with that state. Since the world is often uncertain, utility-based agents choose actions that maximize expected utility, ensuring they make the most favorable decision under uncertain conditions.

UNIT II PROBLEM SOLVING WITH SEARCH TECHNIQUES

Informed Search - Greedy Best First - A* algorithm - Adversarial Game and Search - Game theory - Optimal decisions in game - Min Max Search algorithm - Alpha-beta pruning - Constraint Satisfaction Problems (CSP) - Examples - Map Coloring - Job Scheduling - Backtracking Search for CSP

PART-A

1.what is informed search?

Informed search algorithm uses the idea of heuristic, so it is also called Heuristic search. Heuristic is a function which is used in Informed Search, and it finds the most promising path. It takes the current state of the agent as its input and produces the estimation of how close agent is from the goal.

2. What is greedy best-first search?

Greedy Best-First Search is an AI search algorithm that attempts to find the most promising path from a given starting point to a goal. It prioritizes paths that appear to be the most promising, regardless of whether or not they are actually the shortest path.

3. What is the difference between a * search and greedy best-first search?

Greedy best-first search expands nodes with minimal h(n). It is not optimal, but is often efficient. A* search expands nodes with minimal f(n)=g(n)+h(n). A* s complete and optimal, provided that h(n) is admissible (for TREE-SEARCH) or consistent (for GRAPH-SEARCH).

4.what is adversarial search in game playing?

Adversarial search in artificial intelligence is a problem-solving technique that focuses on making decisions in competitive or adversarial scenarios. It is employed to find optimal strategies when multiple agents, often referred to as players, have opposing or conflicting objectives

5. What is a CSP used for?

A CSP can be used to control the resources that a document is allowed to load. This is primarily used for protection against cross-site scripting (XSS) attacks

6. What is backtracking search for CSP in AI?

Backtracking is a depth-first search algorithm that assigns values to variables one at a time. If an assignment violates a constraint, the algorithm "backtracks" to the previous variable and tries a different value.

7.Write an algorithm for backtracking search for csp?

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
    return BACKTRACK({},csp)
function BACKTRACK(assignment, csp) returns a solution, or failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment then
            add {var = value} to assignment
            inferences ← INFERENCE(csp, var,value)
            if inferences ≠ failure then
                add inferences to assignment
                result ← BACKTRACK(assignment, csp)
                if result ≠ failure then
                    return result
    remove {var = value} and inferences from assignment
return failure
```

8. Define Evaluation function, f(n).

A node with the lowest evaluation is selected for expansion, because evaluation measures distance to the goal.

8. Define Evaluation function, f(n).

h (n) is defined as the estimated cost of the cheapest path from node n to a goal node.

10. Define A* search.

A* search evaluates nodes by combining g(n), the cost to reach the node and h(n), the cost to get from the node to the goal. f(n) = g(n) + h(n)

PART - B

1.Explain Alpha-Beta Pruning?

- Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
- As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called pruning. This involves two threshold parameter Alpha and beta for future expansion, so it is called alpha-beta pruning. It is also called as Alpha-Beta Algorithm.
- Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.

The two-parameter can be defined as:

- Alpha: The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is -∞.
- Beta: The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is +∞.

The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

Condition for Alpha-beta pruning:

The main condition which required for alpha-beta pruning is:

1. $\alpha >= \beta$

Key points about alpha-beta pruning:

- The Max player will only update the value of alpha.
- The Min player will only update the value of beta.
- While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
- We will only pass the alpha, beta values to the child nodes.
- Working of Alpha-Beta Pruning

Let's take an example of two-player search tree to understand the working of Alpha-beta pruning:

Step 1: At the first step the, Max player will start first move from node A where $\alpha = -\infty$ and $\beta = +\infty$, these value of alpha and beta passed down to node B where again $\alpha = -\infty$ and $\beta = +\infty$, and Node B passes the same value to its child D.

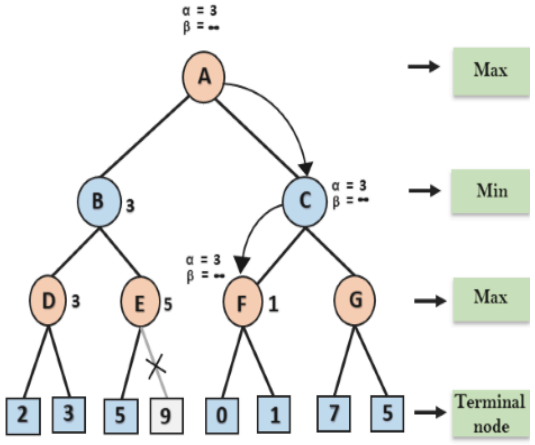
Step 2: At Node D, the value of α will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and node value will also 3.

Step 3: Now algorithm backtrack to node B, where the value of β will change as this is a turn of Min, Now $\beta = +\infty$, will compare with the available subsequent nodes value, i.e. min (∞, 3) = 3, hence at node B now $\alpha = -\infty$, and $\beta = 3$. In the next step, algorithm traverse the next successor of Node B which is node E, and the values of $\alpha = -\infty$, and $\beta = 3$ will also be passed.

Step 4: At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so max (-∞, 5) = 5, hence at node E α= 5 and β= 3, where α>=β, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.

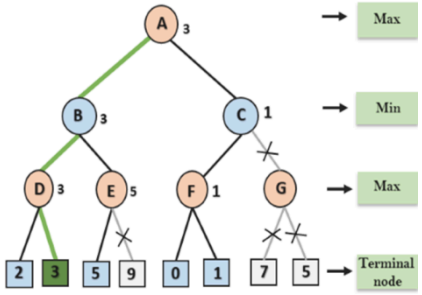
Step 5: At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max (-∞, 3)= 3, and β= +∞, these two values now passes to right successor of A which is Node C. At node C, α=3 and β= +∞, and the same values will be passed on to node F.

Step 6: At node F, again the value of α will be compared with left child which is 0, and max(3,0)= 3, and then compared with right child which is 1, and max(3,1)= 3 still α remains 3, but the node value of F will become 1.



Step 7: Node F returns the node value 1 to node C, at C α= 3 and β= +∞, here the value of beta will be changed, it will compare with 1 so min (∞, 1) = 1. Now at C, α=3 and β= 1, and again it satisfies the condition α>=β, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.

Step 8: C now returns the value of 1 to A here the best value for A is max (3, 1) = 3. Following is the final game tree which is the showing the nodes which are computed and nodes which has never computed. Hence the optimal value for the maximizer is 3 for this example.



2.Explain CSP indetail?

Constraint Satisfaction Problems (CSP) play a crucial role in artificial intelligence (AI) as they help solve various problems that require decision-making under certain constraints. CSPs represent a class of problems where the goal is to find a solution that satisfies a set of

constraints. These problems are commonly encountered in fields like scheduling, planning, resource allocation, and configuration.

Common applications of CSPs include:

- Scheduling: Assigning resources like employees or equipment while respecting time and availability constraints.
- Planning: Organizing tasks with specific deadlines or sequences.
- Resource Allocation: Distributing resources efficiently without overuse

Components of Constraint Satisfaction Problems

CSPs are composed of three key elements:

1. Variables: The things that need to be determined are variables. Variables in a CSP are the objects that must have values assigned to them in order to satisfy a particular set of constraints. Boolean, integer, and categorical variables are just a few examples of the various types of variables, for instance, could stand in for the many puzzle cells that need to be filled with numbers in a sudoku puzzle.

2. Domains: The range of potential values that a variable can have is represented by domains. Depending on the issue, a domain may be finite or limitless. For instance, in Sudoku, the set of numbers from 1 to 9 can serve as the domain of a variable representing a problem cell.

3. Constraints: The guidelines that control how variables relate to one another are known as constraints. Constraints in a CSP define the ranges of possible values for variables. Unary constraints, binary constraints, and higher-order constraints are only a few examples of the various sorts of constraints. For instance, in a sudoku problem, the restrictions might be that each row, column, and 3×3 box can only have one instance of each number from 1 to 9.

Types of Constraint Satisfaction Problems

CSPs can be classified into different types based on their constraints and problem characteristics:

1. Binary CSPs: In these problems, each constraint involves only two variables. For example, in a scheduling problem, the constraint could specify that task A must be completed before task B.

2. Non-Binary CSPs: These problems have constraints that involve more than two variables. For instance, in a seating arrangement problem, a constraint could state that three people cannot sit next to each other.

3. Hard and Soft Constraints: Hard constraints must be strictly satisfied, while soft constraints can be violated, but at a certain cost. This distinction is often used in real-world applications where not all constraints are equally important.

Representation of Constraint Satisfaction Problems (CSP)

In Constraint Satisfaction Problems (CSP), the solution process involves the interaction of variables, domains, and constraints. Below is a structured representation of how CSP is formulated:

1. Finite Set of Variables (V1,V2,…,Vn)(V1,V2,…,Vn):

The problem consists of a set of variables, each of which needs to be assigned a value that satisfies the given constraints.

2. Non-Empty Domain for Each Variable (D1,D2,…,Dn)(D1,D2,…,Dn):

Each variable has a domain—a set of possible values that it can take. For example, in a Sudoku puzzle, the domain could be the numbers 1 to 9 for each cell.

3. Finite Set of Constraints (C1,C2,…,Cm)(C1,C2,…,Cm):

Constraints restrict the possible values that variables can take. Each constraint defines a rule or relationship between variables.

4. Constraint Representation:

Each constraint CiCi is represented as a pair <scope, relation>, where:

- Scope: The set of variables involved in the constraint.
- Relation: A list of valid combinations of variable values that satisfy the constraint.

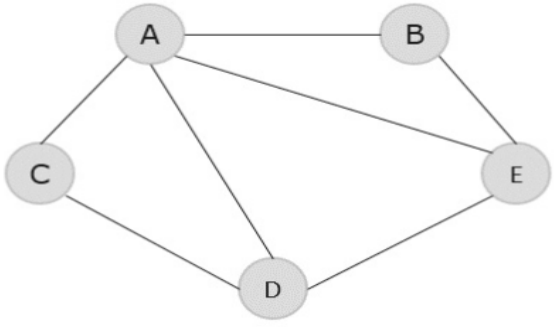
Map Colouring Algorithm

With the map colouring algorithm, a graph G and the colours to be added to the graph are taken as an input and a coloured graph with no two adjacent vertices having the same colour is achieved.

Algorithm

- Initiate all the vertices in the graph.
- Select the node with the highest degree to colour it with any colour.
- Choose the colour to be used on the graph with the help of the selection colour function so that no adjacent vertex is having the same colour.
- Check if the colour can be added and if it does, add it to the solution set.
- Repeat the process from step 2 until the output set is ready.

Example:



Step 1 Find degrees of all the vertices −

A – 4

B – 2

C – 2

D – 3

E – 3

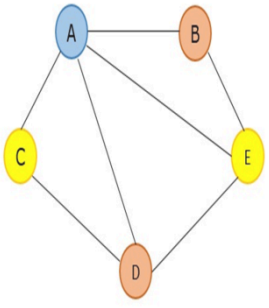
Step 2 Choose the vertex with the highest degree to colour first, i.e., A and choose a colour using selection colour function. Check if the colour can be added to vertex and if yes, add it to the solution set.

Step 3 Select any vertex with the next highest degree from the remaining vertices and colour it using selection colour function. D and E both have the next highest degree 3, so choose any one between them, say D. D is adjacent to A, therefore it cannot be coloured in the same colour as A. Hence, choose a different colour using selection colour function.

Step 4 The next highest degree vertex is E, hence choose E. E is adjacent to both A and D, therefore it cannot be coloured in the same colours as A and D. Choose a different colour using selection colour function.

Step 5 The next highest degree vertices are B and C. Thus, choose any one randomly. B is adjacent to both A and E, thus not allowing to be coloured in the colours of A and E but it is not adjacent to D, so it can be coloured with D's colour.

Step 6 The next and the last vertex remaining is C, which is adjacent to both A and D, not allowing it to be coloured using the colours of A and D. But it is not adjacent to E, so it can be coloured in E's colour.



3.Explain backtracking search for csp in artificial intelligence?

Backtracking is a systematic way to explore all possible solutions by trying each possibility one by one, and if it fails to find a solution, it backtracks to the last valid state and tries the next possibility.

Components:

- Variables: Represent the unknowns to be determined.
- Domains: Set of values each variable can take.
- Constraints: Restrictions on the possible values of combinations of variables.

Algorithm:

- Choose an unassigned variable.
- Try all values in its domain.
- Move to the next variable recursively.
- If a variable's domain is empty without satisfying constraints, backtrack to the previous variable and try another value.

Heuristics:

- Variable ordering: Choose the next variable to assign (e.g., most constrained variable first).
- Value ordering: Choose the order to consider values for a variable (e.g., least constraining value).

Applications:

- Widely used in scheduling problems, puzzles (like Sudoku), and planning.
- Efficient for problems where constraints can greatly reduce the search space.

Optimizations:

- Forward Checking: Prunes the search space by eliminating values that violate constraints.
- Constraint Propagation: Continuously applies constraints to reduce domains of variables.

UNIT III LEARNING

Machine Learning: Definitions – Classification - Regression - approaches of machine learning models - Types of learning - Probability - Basics - Linear Algebra – Hypothesis space and inductive bias, Evaluation. Training and test sets, cross validation, Concept of over fitting, under fitting, Bias and Variance - Regression: Linear Regression - Logistic Regression

PART-A

What is Machine Learning?

Machine learning is a branch of computer science which deals with system programming in order to automatically learn and improve with experience. For example: Robots are programed so that they can perform the task based on data they gather from sensors. It automatically learns programs from data.

2. What is ‘Overfitting’ in Machine learning?

In machine learning, when a statistical model describes random error or noise instead of underlying relationship ‘overfitting’ occurs. When a model is excessively complex, overfitting is normally observed, because of having too many parameters with respect to the number of training data types. The model exhibits poor performance which has been over fit.

3.What are the different Algorithm techniques in Machine Learning?

The different types of techniques in Machine Learning are

- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning
- Transduction
- Learning to Learn

4. What is the main key difference between supervised and unsupervised machine

learning?

Supervised learning	In supervised learning
The supervised learning technique needs labelled data to train the model. For example, to solve a classification problem (a supervised learning task), you need to have label data to train the model and to classify the data into your labelled groups.	Unsupervised learning does not need any labelled dataset. This is the main key difference between supervised learning and unsupervised learning.

5. What is a Linear Regression?

In simple terms, linear regression is adopting a linear approach to modeling the relationship between a dependent variable (scalar response) and one or more independent variables (explanatory variables). In case you have one explanatory variable, you call it a simple linear regression. In case you have more than one independent variable, you refer to the process as multiple linear regressions.

6. What are the advantages of Naïve Bayes?

In Naïve Bayes classifier will converge quicker than discriminative models like logistic regression, so you need less training data. The main advantage is that it can't learn interactions between features.

6. What are the advantages of Naïve Bayes?

It is a flexible and easy-to-use machine learning algorithm that gives great results without even using hyper-parameter tuning. Because of its simplicity and diversity, it is one of the most used algorithms for both classification and regression tasks.

8.Difference between overfitting and Underfitting?

Aspect	Over fitting	Under fitting
Definitio n	Model captures noise and patterns, becoming too complex.	Model fails to capture the underlying trend of the data.
Training Accuracy	Very high	Low
Variance	High variance	Low variance
Examples	Deep neural network fitting small dataset perfectly.	Linear regression on non-linear data.

9. What is the difference between classification and regression?

Classification is used to produce discrete results; classification is used to classify data into some specific categories. For example, classifying emails into spam and non-spam categories. Whereas, we use regression analysis when we are dealing with continuous data, for example predicting stock prices at a certain point in time.

10. What are types of classification models?

- Logistic Regression
- Naïve Bayes
- K-Nearest Neighbors
- Decision Tree
- Support Vector Machines

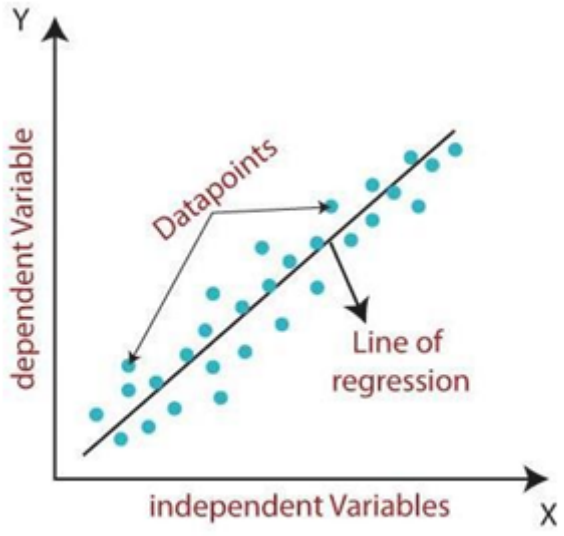
PART –B

1.Explain Linear Regression in Machine Learning ?

• Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.

• Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

• The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:



The values for x and y variables are training datasets for Linear Regression model representation.

Types of Linear Regression

Linear regression can be further divided into two types of the algorithm:

- SimpleLinearRegression:

If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.

If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression. Linear Regression Line A linear line showing the relationship between the dependent and independent variables is called a regression line. A regression line can show two types of relationship: o PositiveLinearRelationship: If the dependent variable increases on the Y-axis and independent variable increases on X- axis, then such a relationship is termed as a Positive linear relationship. o NegativeLinearRelationship: If the dependent variable decreases on the Y-axis and independent variable increases on the X- axis, then such a relationship is called a negative linear relationship. Finding the best fit line:

When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error. The different values for weights or the coefficient of lines (a0, a1) gives a different line of regression, so we need to calculate the best values for a0 and a1 to find the best fit line, so to calculate this we use cost function. Cost function- o The different values for weights or coefficient of lines (a0, a1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line. o Cost function optimizes the regression coefficients or weights. It measures how a linear regression model is performing. o We can use the cost function to find the accuracy of the mapping function, which maps the input variable to the output variable. This mapping function is also known as Hypothesis function. For Linear Regression, we use the Mean Squared Error (MSE) cost function, which is the average of squared error occurred between the predicted values and actual values. It can be written as:

$$MSE = 1\frac{1}{N}\sum_{i=1}^{n}(y_i - (a_1x_i + a_0))^2$$

Where,
N=Total number of observation
Yi = Actual value
$(a1x_i+a_0)$= Predicted value.

2.Explain machine learnings model?

Machine learning is a subset of AI, which enables the machine to automatically learn from data, improve performance from past experiences, and make predictions. Machine learning contains a set of algorithms that work on a huge amount of data. Data is fed to these algorithms to train them, and on the basis of training, they build the model & perform a specific task.

These ML algorithms help to solve different business problems like Regression, Classification, Forecasting, Clustering, and Associations, etc. Based on the methods and way of learning, machine learning is divided into mainly four types, which are:

1. Supervised Machine Learning

2. Unsupervised Machine Learning

3. Semi-Supervised Machine Learning

4. Reinforcement Learning

1. Supervised Machine Learning

Supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Example: Suppose we have an input dataset of cats and dog images. So, first, we will provide the training to the machine to understand the images, such as the shape & size of the tail of cat and dog, Shape of eyes, colour, height (dogs are taller, cats are smaller), etc. After completion of training, we input the picture of a cat and ask the machine to identify the object and predict the output. Now, the machine is well trained, so it will check all the features of the object, such as height, shape, color, eyes, ears, tail, etc., and find that it's a cat. So, it will put it in the Cat category.

This is the process of how the machine identifies the objects in Supervised Learning.

The main goal of the supervised learning technique is to map the input variable(x) with the output variable(y). Some real-world applications of supervised learning are Risk Assessment, Fraud Detection, Spam filtering, etc.

Supervised machine learning can be classified into two types of problems, which are given below:

- Classification
- Regression

a) Classification Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as "Yes" or No, Male or Female, Red or Blue, etc.

b) The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are Spam Detection, Email filtering, etc. Some popular classification algorithms are given below:

- Random Forest Algorithm
- Decision Tree Algorithm
- Logistic Regression Algorithm
- Support Vector Machine Algorithm

c) Regression

Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables. These are used to predict continuous output variables, such as market trends, weather prediction, etc.

Some popular Regression algorithms are given below:

- Simple Linear Regression Algorithm o Multivariate Regression Algorithm
- Decision Tree Algorithm o Lasso Regression Advantages

Since supervised learning work with the labelled dataset so we can have an exact idea about the classes of objects. These algorithms are helpful in predicting the output on the basis of prior experience.

2. Unsupervised Machine Learning

Unsupervised learning is different from the Supervised learning technique; as its name suggests, there is no need for supervision. It means, in unsupervised machine learning, the machine is trained using the unlabeled dataset, and the machine predicts the output without any supervision. In unsupervised learning, the models are trained with the data that is neither classified nor labelled, and the model acts on that data without any supervision.

Unsupervised Learning can be further classified into two types, which are given below:

- Clustering
- Association

Advantages:

- These algorithms can be used for complicated tasks compared to the supervised ones because these algorithms work on the unlabeled dataset.
- Unsupervised algorithms are preferable for various tasks as getting the unlabeled dataset is easier as compared to the labelled dataset.

Disadvantages:

- The output of an unsupervised algorithm can be less accurate as the dataset is not labelled, and algorithms are not trained with the exact output in prior.
- Working with Unsupervised learning is more difficult as it works with the unlabelled dataset that does not map with the output.

Applications of Unsupervised Learning

- Network Analysis
- Recommendation Systems

Unit IV

SUPERVISED LEARNING

Neural Network: Introduction, Perceptron Networks – Adaline – Back propagation networks - Decision Tree: Entropy – Information gain – Gini Impurity – classification algorithm – Rule based Classification – Naive Bayesian classification – Support Vector Machines (SVM)

What is bagging and boosting in ensemble learning?

Bagging is a way to decrease the variance in the prediction by generating additional data for training from dataset using combinations with repetitions to produce multi-sets of the original data. Boosting is an iterative technique which adjusts the weight of an observation based on the last classification.

2. List the Ensemble Techniques.

A few simple but powerful techniques namely:

- Max Voting
- Averaging
- Weighted Averaging

3. What are the Advanced Ensemble Techniques?

- Bagging
- Boosting

4. What is stacking in ensemble learning?

Stacking is one of the most popular ensemble machine learning techniques used to predict multiple nodes to build a new model and improve model performance. Stacking enables us to train multiple models to solve similar problems, and based on their combined output, it builds a new model with improved performance.

5. Which are the three types of ensemble learning?

The three main classes of ensemble learning methods are bagging, stacking, and boosting, and it is important to both have a detailed understanding of each method and to consider them on your predictive modeling project.

6. Why ensemble methods are used?

There are two main reasons to use an ensemble over a single model, and they are related; they are: Performance: An ensemble can make better predictions and achieve better performance than any single contributing model. Robustness: An ensemble reduces thespread or dispersion of the predictions and model performance.

7. What is a voting classifier?

A voting classifier is a machine learning estimator that trains various base models or estimators and predicts on the basis of aggregating the findings of each base estimator. The aggregating criteria can be combined decision of voting for each estimator output

8. What is a voting classifier?

The performance-weighted-voting model integrates five classifiers including logistic regression, SVM, random forest, XG Boost and neural networks. We first used cross-validation to get the predicted results for the five classifiers.

9. What is difference between K means and Gaussian mixture?

K-Means is a simple and fast clustering method, but it may not truly capture heterogeneity inherent in Cloud workloads. Gaussian Mixture Models can discover complex patterns and group them into cohesive, homogeneous components that are close representatives of real patterns within the data set.

10. What are Gaussian mixture models? How is expectation maximization used in it?

Expectation maximization provides an iterative solution to maximum likelihood estimation with latent variables. Gaussian mixture models are an approach to density estimation where the parameters of the distributions are fit using the expectation-maximization algorithm.

11.What is ‘Training set’ and ‘Test set’?

The training set trains the machine learning model, allowing it to learn the patterns and relationships within the data. The test set used after the model has been trained and validated, to provide an unbiased evaluation of the model performance on completely new, unseen data.

12.Given that P(A)=0.3,P(A|B)=0.4 and P(B)=0.5, Compute P(B|A).

P(A/B)=[P(A)*P(B/A)]/P(B)

0.4 = (0.3*P(B/A))/0.5

P(B/A) = 0.66

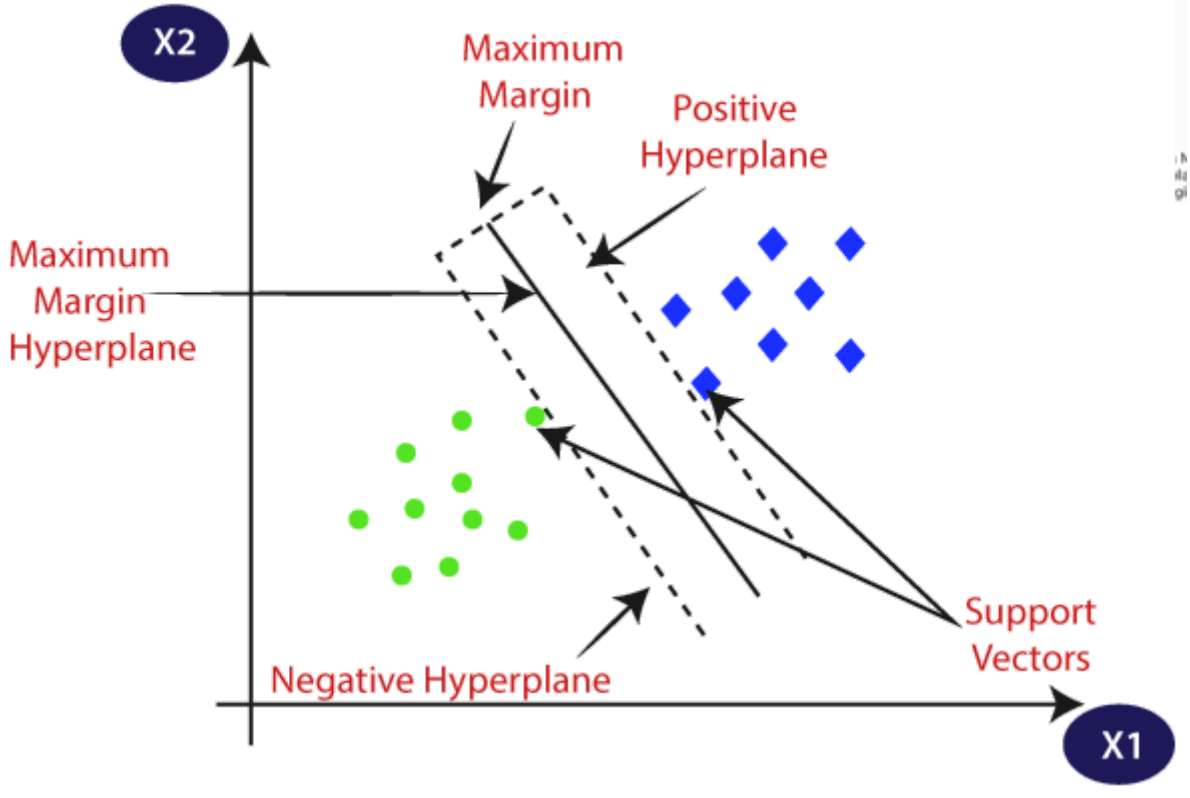
PART –B

1. Explain SVM Algorithm in Detail

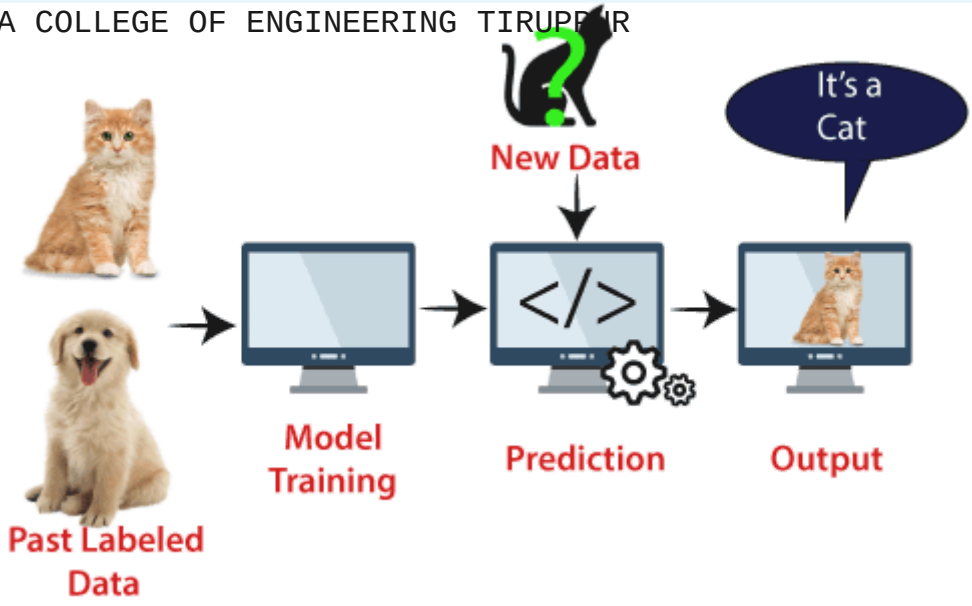
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

Types of SVM

SVM can be of two types

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

2. Explain Naïve Bayes Classifier with an Example.

Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.

- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**

Bayes' Theorem:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for **text classification problems**.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

Applications of Naïve Bayes Classifier:

- It is used for **Credit Scoring**.
- It is used in **medical data classification**.
- It can be used in **real-time predictions** because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as **Spam filtering** and **Sentiment analysis**.

Types of Naïve Bayes Model:

There are three types of Naive Bayes Model, which are given below:

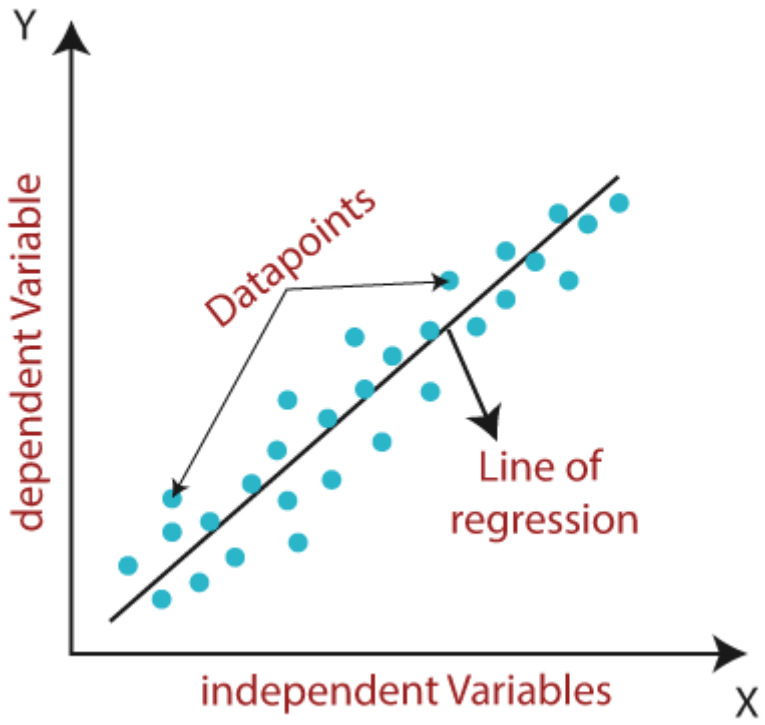
- **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc.
  The classifier uses the frequency of words for the predictors.
- **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

3. Explain the following a) Linear regression b) Logistic Regression

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price,** etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:



$= a_0 + a_1 x + \varepsilon$

Here,

Y= Dependent Variable (Target Variable)
X= Independent Variable (predictor Variable)
a0= intercept of the line (Gives an additional degree of freedom)
a1 = Linear regression coefficient (scale factor to each input value).
ε = random error

The values for x and y variables are training datasets for Linear Regression model representation.

Types of Linear Regression

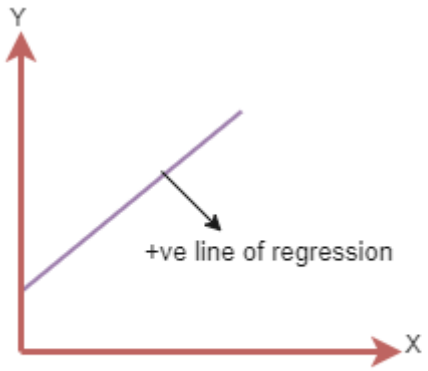
Linear regression can be further divided into two types of the algorithm:

- **Simple Linear Regression:**
  If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.
- **Multiple Linear regression:**
  If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

Linear Regression Line

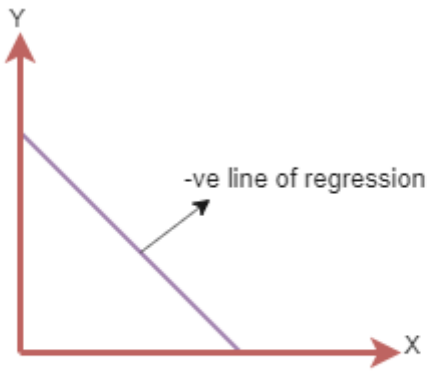
A linear line showing the relationship between the dependent and independent variables is called a **regression line**. A regression line can show two types of relationship:

- **Positive Linear Relationship:**
  If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.



The line equation will be: $Y= a_0+a_1x$

- **Negative Linear Relationship:**
  If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.



The line of equation will be: $Y= -a_0+a_1x$

Logistic regression is a **supervised machine learning algorithm** used for **classification tasks** where the goal is to predict the probability that an instance belongs to a given class or not. Logistic regression is a statistical algorithm which analyze the relationship between two data factors. The article explores the fundamentals of logistic regression, it's types and

implementations. Logistic regression is used for binary classification where we use sigmoid function, that takes input as independent variables and produces a probability value between 0 and 1.

For example, we have two classes Class 0 and Class 1 if the value of the logistic function for an input is greater than 0.5 (threshold value) then it belongs to Class 1 otherwise it belongs to Class 0. It's referred to as regression because it is the extension of linear regression but is mainly used for classification problems.

Types of Logistic Regression

On the basis of the categories, Logistic Regression can be classified into three types:

1. **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
2. **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
3. **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

Differences Between Linear and Logistic Regression

The difference between linear regression and logistic regression is that linear regression output is the continuous value that can be anything while logistic regression predicts the probability that an instance belongs to a given class or not.

Linear Regression	Logistic Regression
Linear regression is used to predict the continuous dependent variable using a given set of independent variables.	Logistic regression is used to predict the categorical dependent variable using a given set of independent variables.
Linear regression is used for solving regression problem.	It is used for solving classification problems.
In this we predict the value of continuous variables	In this we predict values of categorical variables
In this we find best fit line.	In this we find S-Curve.
Least square estimation method is used for estimation of accuracy.	Maximum likelihood estimation method is used for Estimation of accuracy.
The output must be continuous value, such as price, age, etc.	Output must be categorical value such as 0 or 1, Yes or no, etc.
It required linear relationship between dependent and independent variables.	It not required linear relationship.

There may be collinearity between the independent variables.	There should not be collinearity between independent variables.

4. Explain briefly about unsupervised learning structure?

Unsupervised machine learning is the process of inferring underlying hidden patterns from historical data. Within such an approach, a machine learning model tries to find any similarities, differences, patterns, and structure in data by itself. No prior human intervention is needed.

Unsupervised learning finds a myriad of real-life applications, including:

- data exploration,
- customer segmentation,
- recommender systems,
- target marketing campaigns, and
- data preparation and visualization, etc.

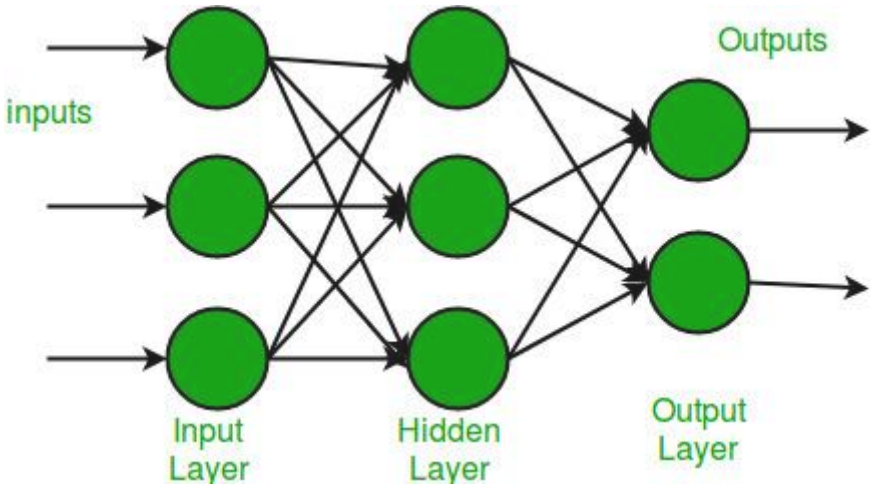
5. Draw the architecture of a Multilayer perceptron (MLP) and explain its operation. Mention its advantages and disadvantages?

Multi-layer perception is also known as MLP. It is fully connected dense layers, which transform any input dimension to the desired dimension. A multi-layer perception is a neural network that has multiple layers. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons.

A gentle introduction to **neural networks and TensorFlow** can be found here:

- Neural Networks
- Introduction to TensorFlow

A multi-layer perceptron has one input layer and for each input, there is one neuron(or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes. A schematic diagram of a Multi-Layer Perceptron (MLP) is depicted below.



In the multi-layer perceptron diagram above, we can see that there are three inputs and thus three input nodes and the hidden layer has three nodes. The output layer gives two outputs, therefore there are two output nodes. The nodes in the input layer take input and forward it for further process, in the diagram above the nodes in the input layer forwards their output to

each of the three nodes in the hidden layer, and in the same way, the hidden layer processes the information and passes it to the output layer.

Every node in the multi-layer perception uses a sigmoid activation function. The sigmoid activation function takes real values as input and converts them to numbers between 0 and 1 using the sigmoid formula. Now that we are done with the theory part of multi-layer perception, let's go ahead and implement some code in **python** using the **TensorFlow** library.

6. List the factors that affect the performance of multilayer feed-forward neural network.

Feedforward Neural Networks are artificial neural networks where the node connections do not form a cycle. They are biologically inspired algorithms that have several neurons like units arranged in layers. The units in neural networks are connected and are called nodes. Data enters the network at the point of input, seeps through every layer before reaching the output. However, the connections differ in strength or weight. The weight of the connections provides vital information about a network.

Feedforward Neural Networks are also known as multi-layered networks of neurons (MLN). The neuron network is called feedforward as the information flows only in the forward direction in the network through the input nodes. There is no feedback connection so that the network output is fed back into the network without flowing out.

These networks are depicted through a combination of simple models, known as sigmoid neurons. The sigmoid neuron is the foundation for a feedforward neural network.

The feedforward neural networks comprise the following components:

- Input layer
- Output layer
- Hidden layer
- Neuron weights
- Neurons
- Activation function

Input layer: This layer comprises neurons that receive the input and transfer them to the different layers in the network. The number of neurons in the input layer must be the same as the number of the features or attributes in the dataset.

Output layer: This layer is the forecasted feature that depends on the type of model being built.

Hidden layer: The hidden layers are positioned between the input and the output layer. The number of hidden layers depends on the type of model. Hidden layers have several neurons that impose transformations on the input before transferring. The weights in the network are constantly updated to make it easily predictable.

Neuron weights: The strength or the magnitude of connection between two neurons is called weights. The input weights can be compared just as coefficients in linear regression. The value of the weights is usually small and falls within the range of 0 to 1.

Neurons: The feedforward network has artificial neurons, which are an adaptation of biological neurons. Artificial neurons are the building blocks of the neural network. The neurons work in two ways: first, they determine the sum of the weighted inputs, and, second, they initiate an activation process to normalize the sum.

The activation function can be either linear or nonlinear. Weights are related to each input of the neuron. The network studies these weights during the learning phase.

Activation Function: This is the decision-making center at the neuron output. The neurons finalize linear or non-linear decisions based on the activation function. It prevents the enlargement of neuron outputs due to cascading effect because of passing through many layers. The three most important activation functions are sigmoid, Tanh, and Rectified Linear Unit (ReLu).

- **Sigmoid:** It maps the input values within the range of 0 to 1.
- **Tanh:** It maps the input values between -1 and 1.
- **Rectified linear Unit**: This function allows only the positive values to flow through. The negative values are mapped at 0.

Unit V

UNSUPERVISED LEARNING

Unsupervised Learning – Principle Component Analysis – Neural Network: Fixed Weight

Competitive Nets – Kohonen Self-Organizing Feature Maps – Clustering: Definition – Types

of Clustering – Hierarchical clustering algorithms – k-means algorithm

1. What is k-means unsupervised learning?

K-Means clustering is an unsupervised learning algorithm. There is no labeled data for this clustering, unlike in supervised learning. K-Means performs the division of objects into clusters that share similarities and are dissimilar to the objects belonging to another cluster. The term 'K' is a number.

2. What is the difference between K-means and KNN?

KNN is a supervised learning algorithm mainly used for classification problems, whereas K-Means (aka K-means clustering) is an unsupervised learning algorithm. K in K-Means refers to the number of clusters, whereas K in KNN is the number of nearest neighbors(based on the chosen distance metric).

3. State the applications of K-Means Clustering.

K-Means clustering is used in a variety of examples or business cases in real life, like:

- Academic performance
- Diagnostic systems
- Search engines
- Wireless sensor networks

4. List some of the popular unsupervised learning algorithms.

- K-means clustering
- KNN (k-nearest neighbors)
- Hierarchal clustering
- Anomaly detection
- Neural Networks
- Principle Component Analysis
- Independent Component Analysis

- Apriori algorithm
- Singular value decomposition

5.What is expectation maximization algorithm used for?

`The EM algorithm is used to find (local) maximum likelihood parameters of a statistical model in cases where the equations cannot be solved directly. Typically these models involve latent variables in addition to unknown parameters and known data observations.

6. What is the advantage of Gaussian process?

Gaussian processes are a powerful algorithm for both regression and classification. Their greatest practical advantage is that they can give a reliable estimate of their own uncertainty.

7. What are examples of unsupervised learning?

Some examples of unsupervised learning algorithms include K-Means Clustering, Principal Component Analysis and Hierarchical Clustering.

8. Mention the pros and cons of KNN Algorithm.

Advantages:

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:

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.

9. How do you implement expectation maximization algorithm?

The two steps of the EM algorithm are:

1) E-step: perform probabilistic assignments of each data point to some class based on the current hypothesis h for the distributional class parameters;

2) M-step: update the hypothesis h for the distributional class parameters based on the new data assignments.

10. What is the principle of maximum likelihood?

The principle of maximum likelihood is a method of obtaining the optimum values of the parameters that define a model. And while doing so, you increase the likelihood of your model reaching the “true” model.

11. What is Bayesian Belief Network?

A Bayesian network is a probabilistic graphical model that represents a set of variables and their conditional dependencies via a directed acyclic graph. While it is one of several forms of causal notation, causal networks are special cases of Bayesian networks.

12. What is difference between K means and Gaussian mixture?

K-Means is a simple and fast clustering method, but it may not truly capture heterogeneity inherent in Cloud workloads. Gaussian Mixture Models can discover complex patterns and group them into cohesive, homogeneous components that are close representatives of real patterns within the data set.

13. What is Random forest?

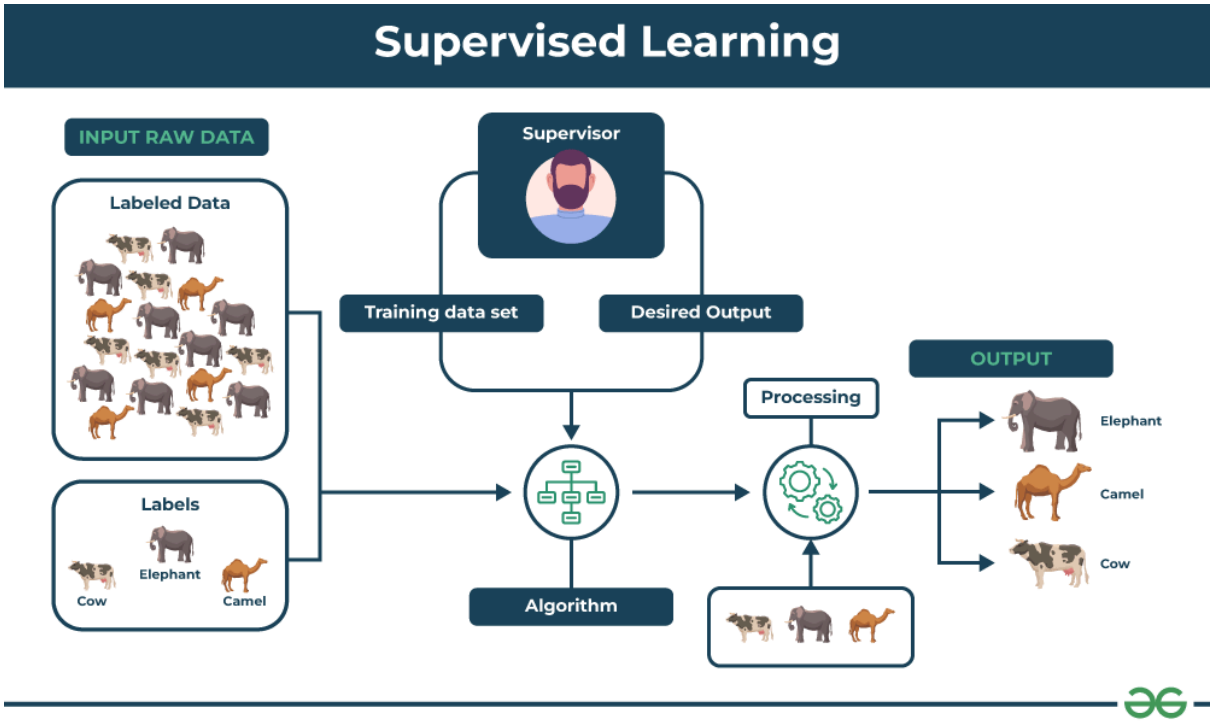
Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees

PART –B

1. Explain various learning techniques involved in unsupervised learning?

Supervised learning, as the name indicates, has the presence of a supervisor as a teacher. Supervised learning is when we teach or train the machine using data that is well-labelled. Which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples(data) so that the supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labeled data.

For example, a labeled dataset of images of Elephant, Camel and Cow would have each image tagged with either “Elephant” , “Camel”or “Cow.”



Key Points:

- Supervised learning involves training a machine from labeled data.
- Labeled data consists of examples with the correct answer or classification.
- The machine learns the relationship between inputs (fruit images) and outputs (fruit labels).
- The trained machine can then make predictions on new, unlabeled data.

Types of Supervised Learning

Supervised learning is classified into two categories of algorithms:

- **Regression**: A regression problem is when the output variable is a real value, such as "dollars" or "weight".
- **Classification**: A classification problem is when the output variable is a category, such as "Red" or "blue" , "disease" or "no disease".

Supervised learning deals with or learns with "labeled" data. This implies that some data is already tagged with the correct answer.

1- Regression

Regression is a type of supervised learning that is used to predict continuous values, such as house prices, stock prices, or customer churn. Regression algorithms learn a function that maps from the input features to the output value.

Some common regression algorithms include:

- Linear Regression
- Polynomial Regression
- Support Vector Machine Regression
- Decision Tree Regression
- Random Forest Regression

2- Classification

Classification is a type of supervised learning that is used to predict categorical values, such as whether a customer will churn or not, whether an email is spam or not, or whether a medical image shows a tumor or not. Classification algorithms learn a function that maps from the input features to a probability distribution over the output classes.

Some common classification algorithms include:

- Logistic Regression
- Support Vector Machines
- Decision Trees
- Random Forests
- Naive Baye

Evaluating Supervised Learning Models

Evaluating supervised learning models is an important step in ensuring that the model is accurate and generalizable. There are a number of different metrics that can be used to evaluate supervised learning models, but some of the most common ones include:

For Regression

- **Mean Squared Error (MSE):** MSE measures the average squared difference between the predicted values and the actual values. Lower MSE values indicate better model performance.
- **Root Mean Squared Error (RMSE):** RMSE is the square root of MSE, representing the standard deviation of the prediction errors. Similar to MSE, lower RMSE values indicate better model performance.

- **Mean Absolute Error (MAE):** MAE measures the average absolute difference between the predicted values and the actual values. It is less sensitive to outliers compared to MSE or RMSE.
- **R-squared (Coefficient of Determination):** R-squared measures the proportion of the variance in the target variable that is explained by the model. Higher R-squared values indicate better model fit.

For Classification

- **Accuracy:** Accuracy is the percentage of predictions that the model makes correctly. It is calculated by dividing the number of correct predictions by the total number of predictions.
- **Precision:** Precision is the percentage of positive predictions that the model makes that are actually correct. It is calculated by dividing the number of true positives by the total number of positive predictions.
- **Recall:** Recall is the percentage of all positive examples that the model correctly identifies. It is calculated by dividing the number of true positives by the total number of positive examples.
- **F1 score:** The F1 score is a weighted average of precision and recall. It is calculated by taking the harmonic mean of precision and recall.
- **Confusion matrix:** A confusion matrix is a table that shows the number of predictions for each class, along with the actual class labels. It can be used to visualize the performance of the model and identify areas where the model is struggling.

Applications of Supervised learning

Supervised learning can be used to solve a wide variety of problems, including:

- **Spam filtering:** Supervised learning algorithms can be trained to identify and classify spam emails based on their content, helping users avoid unwanted messages.
- **Image classification:** Supervised learning can automatically classify images into different categories, such as animals, objects, or scenes, facilitating tasks like image search, content moderation, and image-based product recommendations.
- **Medical diagnosis:** Supervised learning can assist in medical diagnosis by analyzing patient data, such as medical images, test results, and patient history, to identify patterns that suggest specific diseases or conditions.
- **Fraud detection:** Supervised learning models can analyze financial transactions and identify patterns that indicate fraudulent activity, helping financial institutions prevent fraud and protect their customers.
- **Natural language processing (NLP):** Supervised learning plays a crucial role in NLP tasks, including sentiment analysis, machine translation, and text summarization, enabling machines to understand and process human language effectively.

Advantages of Supervised learning

- Supervised learning allows collecting data and produces data output from previous experiences.
- Helps to optimize performance criteria with the help of experience.
- Supervised machine learning helps to solve various types of real-world computation problems.
- It performs classification and regression tasks.

- It allows estimating or mapping the result to a new sample.
- We have complete control over choosing the number of classes we want in the training data.

Disadvantages of Supervised learning

- Classifying big data can be challenging.
- Training for supervised learning needs a lot of computation time. So, it requires a lot of time.
- Supervised learning cannot handle all complex tasks in Machine Learning.
- Computation time is vast for supervised learning.
- It requires a labelled data set.
- It requires a training process.

2. Explain in detail about K-Means clustering algorithm with a suitable example?

K-Means clustering is an unsupervised iterative clustering technique.

- It partitions the given data set into k predefined distinct clusters.
- A cluster is defined as a collection of data points exhibiting certain similarities.

It partitions the data set such that-

- Each data point belongs to a cluster with the nearest mean.
- Data points belonging to one cluster have high degree of similarity.
- Data points belonging to different clusters have high degree of dissimilarity.

K-Means Clustering Algorithm

K-Means Clustering Algorithm involves the following steps-

Step-01:

- Choose the number of clusters K.

Step-02:

- Randomly select any K data points as cluster centers.
- Select cluster centers in such a way that they are as farther as possible from each other.

Step-03:

- Calculate the distance between each data point and each cluster center.
- The distance may be calculated either by using given distance function or by using euclidean distance formula.

Step-04:

- Assign each data point to some cluster.
- A data point is assigned to that cluster whose center is nearest to that data point.

Step-05:

- Re-compute the center of newly formed clusters.

- The center of a cluster is computed by taking mean of all the data points contained in that cluster.

Step-06:

Keep repeating the procedure from Step-03 to Step-05 until any of the following stopping criteria is met-

- Center of newly formed clusters do not change
- Data points remain present in the same cluster
- Maximum number of iterations are reached

Advantages-

K-Means Clustering Algorithm offers the following advantages-

Point-01

It is relatively efficient with time complexity O(nkt) where-

- n = number of instances
- k = number of clusters
- t = number of iterations

Point-02:

- It often terminates at local optimum.
- Techniques such as Simulated Annealing or **Genetic Algorithms** may be used to find the global optimum.

Disadvantages-

K-Means Clustering Algorithm has the following disadvantages-

- It requires to specify the number of clusters (k) in advance.
- It can not handle noisy data and outliers.
- It is not suitable to identify clusters with non-convex shapes.

3. Explain in detail about clustering and list out their types?

Clustering is a type of unsupervised learning method of machine learning. In the unsupervised learning method, the inferences are drawn from the data sets which do not contain labelled output variable. It is an exploratory data analysis technique that allows us to analyze the multivariate data sets.

Clustering is a task of dividing the data sets into a certain number of clusters in such a manner that the data points belonging to a cluster have similar characteristics. Clusters are nothing but the grouping of data points such that the distance between the data points within the clusters is minimal. Clustering is done to segregate the groups with similar traits.

1. Density-Based Clustering
2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
3. OPTICS (Ordering Points to Identify Clustering Structure)

4. HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise)
5. Hierarchical Clustering
6. Fuzzy Clustering
7. Partitioning Clustering
8. PAM (Partitioning Around Medoids)
9. Grid-Based Clustering

4. What is the main key difference between supervised and unsupervised machine learning?

The distinction between supervised and unsupervised learning depends on whether the learning algorithm uses pattern-class information. Supervised learning assumes the availability of a teacher or supervisor who classifies the training examples, whereas unsupervised learning must identify the pattern-class information as a part of the learning process.

Supervised learning algorithms utilize the information on the class membership of each training instance. This information allows supervised learning algorithms to detect pattern misclassifications as feedback to themselves. In unsupervised learning algorithms, unlabeled instances are used. They blindly or heuristically process them. Unsupervised learning algorithms often have less computational complexity and less accuracy than supervised learning algorithms. \

	Supervised Learning	**Unsupervised Learning**
Input Data	Uses Known and Labeled Data as input	Uses Unknown Data as input
Computational Complexity	Less Computational Complexity	More Computational Complex
Real-Time	Uses off-line analysis	Uses Real-Time Analysis of Data
Number of Classes	The number of Classes is known	The number of Classes is not known
Accuracy of Results	Accurate and Reliable Results	Moderate Accurate and Reliable Results
Output data	The desired output is given.	The desired, output is not given.

Model	In supervised learning it is not possible to learn larger and more complex models than in unsupervised learning	In unsupervised learning it is possible to learn larger and more complex models than in supervised learning
Training data	In supervised learning training data is used to infer model	In unsupervised learning training data is not used.
Another name	Supervised learning is also called classification.	Unsupervised learning is also called clustering.
Test of model	We can test our model.	We can not test our model.
Example	Optical Character Recognition	Find a face in an image.

5. Describe in detail about Adaline architecture?

ADALINE (Adaptive Linear Neuron) is an artificial neural network model proposed by Bernard Widrow and Ted Hoff in 1960. It is similar to the perceptron, but instead of a step activation function, it uses a linear activation function.

ADALINE is a supervised learning model used to perform binary classification and linear regression. The neural network consists of an input layer, an output layer and a feedback layer that adjusts the weights of the input layer according to the output obtained.

The objective of ADALINE is to minimise the mean square error (MSE) between the desired output and the actual output of the network. It does this by using the gradient descent algorithm to adjust the input layer weights.

ADALINE is a linear model, which means that it can only learn linear relationships between inputs and outputs. However, it can be used as a basic unit in more complex neural network models, such as multilayer neural networks.

ADALINE (Adaptive Linear Neuron) is an artificial neural network model proposed by Bernard Widrow and Ted Hoff in 1960. It is similar to the perceptron, but instead of a step activation function, it uses a linear activation function.

Adaptive Linear Neuron Learning algorithm

Step 0: initialize the weights and the bias are set to some random values but not to zero, also initialize the learning rate α.

Step 1 − perform steps 2-7 when stopping condition is false.

Step 2 − perform steps 3-5 for each bipolar training pair s:t.

Step 3 − Activate each input unit as follows −

Step 4 − Obtain the net input with the following relation −

Here ‘b’ is bias and ‘n’ is the total number of input neurons.

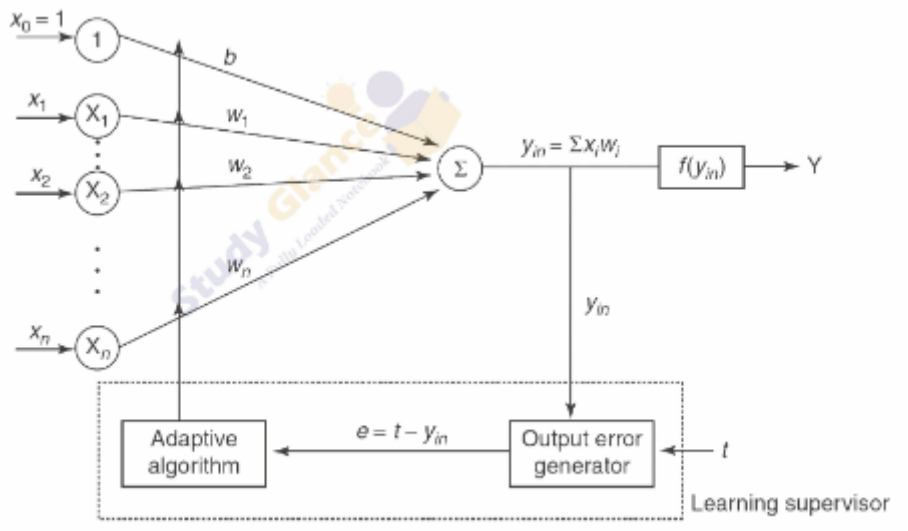
Step 5 Until least mean square is obtained **(t - yin)**, Adjust the weight and bias as follows −

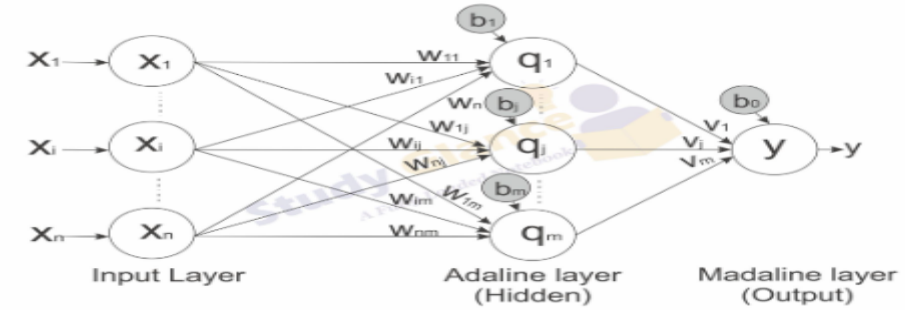
wi(new) = wi(old) + α(t - yin)xi

b(new) = b(old) + α(t - yin)

Now calculate the error using => **E = (t - yin)2**

Step 7 − Test for the stopping condition, if error generated is less then or equal to specified tolerance then stop.





OCS351 – ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FUNDAMENTALS

UNIT III LEARNING

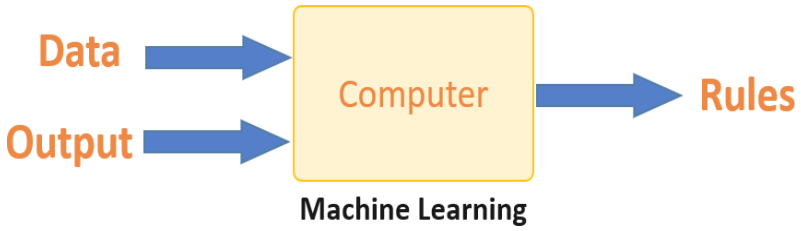
Machine Learning: Definitions – Classification - Regression - approaches of machine learning models - Types of learning - Probability - Basics - Linear Algebra – Hypothesis space and inductive bias, Evaluation. Training and test sets, cross validation, Concept of over fitting, under fitting, Bias and Variance - Regression: Linear Regression – Logistic Regression

I. MACHINE LEARNING: DEFINITIONS

What is machine learning? Need – History and Definitions - Applications

→ Machine Learning is a branch of Artificial Intelligence that allows machines to learn and improve from experience automatically.

→ It is defined as the field of study that gives computers the capability to learn without being explicitly programmed. It is quite different than traditional programming.



→ **AI** (Artificial Intelligence) is a machine's ability to perform cognitive functions as humans do, such as perceiving, learning, reasoning, and solving problems. The benchmark for AI is the human level concerning in teams of reasoning, speech, and vision.

→ Machine learning is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations.

The life of Machine Learning programs is straightforward and can be summarized in the following points:

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 4–7 until the results are satisfying

TYPES OF MACHINE LEARNING

Types of machine learning?

→ Classical machine learning is often categorized by how an algorithm learns to become more accurate in its predictions.There are four basic approaches: supervised learning, unsupervised learning, semi–supervised learning and reinforcement learning.

→ The type of algorithm data scientists choose to use depends on what type of data they want to predict.

- **Supervised learning**
  - → In this type of machine learning, data scientists supply algorithms with labeled training data and define the variables they want the algorithm to assess for correlations.
  - → Both the input and the output of the algorithm is specified.
- **Unsupervised learning**
  - → This type of machine learning involves algorithms that train on unlabeled data.
  - → The algorithm scans through data sets looking for any meaningful connection. The data that algorithms train on as well as the predictions or recommendations they output are predetermined.
- **Semi-supervised learning**
  - → This approach to machine learning involves a mix of the two preceding types. Data scientists may feed an algorithm mostly labeled training data, but the model is free to explore the data on its own and develop its own understanding of the data set.
- **Reinforcement learning**
  - → Data scientists typically use reinforcement learning to teach a machine to complete a multi–step process for which there are clearly defined rules.
  - → Data scientists program an algorithm to complete a task and give it positive or negative cues as it works out how to complete a task. But for the most part, the algorithm decides on its own what steps to take along the way.

How does supervised machine learning work?: Supervised machine learning requires the data scientist to train the algorithm with both labeled inputs and desired outputs. Supervised learning algorithms are good for the following tasks:

- **Binary classification:** Dividing data into two categories.
- **Multi-class classification:** Choosing between more than two types of answers.
- **Regression modeling:** Predicting continuous values.
- **Ensembling:** Combining the predictions of multiple machine learning models to produce an accurate prediction.

II. CLASSIFICATION

Classification

→ As the name suggests, Classification is the task of "classifying things" into sub− categories. But, by a machine! If that doesn't sound like much, imagine your computer being able to differentiate between you and a stranger. Between a potato and a tomato. Between an A grade and an F. Now, it sounds interesting now.

→ In Machine Learning and Statistics, Classification is the problem of identifying to which of a set of categories (subpopulations), a new observation belongs, on the basis of a training set of data containing observations and whose categories membership is known.

Types of Classification

Classification is of two types:

- **Binary Classification**: When we have to categorize given data into 2 distinct classes. Example – On the basis of given health conditions of a person, we have to determine whether the person has a certain disease or not.
- **Multiclass Classification**: The number of classes is more than 2. For Example – On the basis of data about different species of flowers, we have to determine which specie our observation belongs.

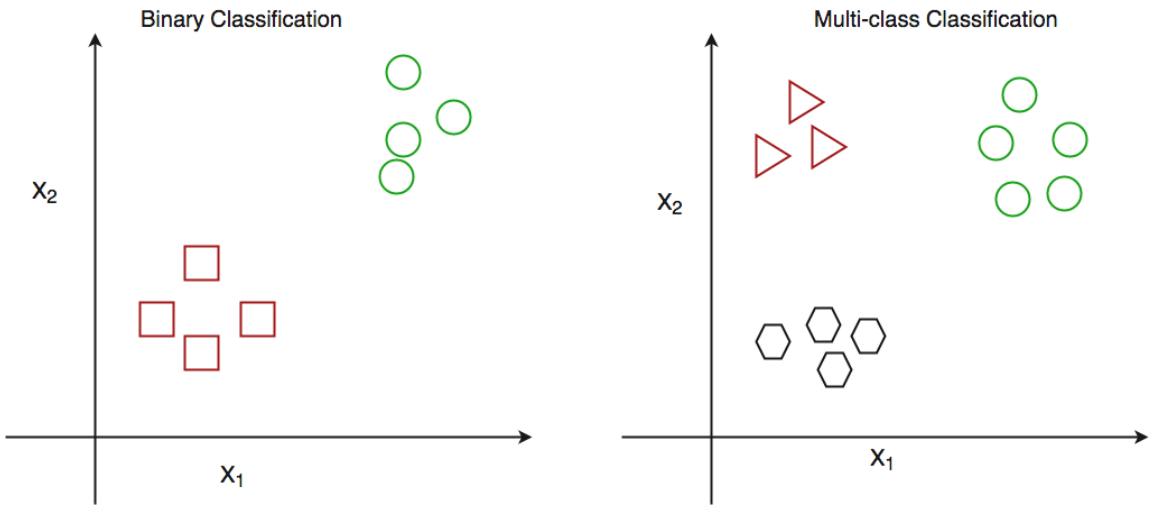


Fig: Binary and Multiclass Classification. Here x1 and x2 are the variables upon which the class is predicted.

How does classification works?

→ Suppose we have to predict whether a given patient has a certain disease or not, on the basis of 3 variables, called features.

→ This means there are two possible outcomes:

- The patient has the said disease. Basically, a result labeled "Yes" or "True".
- The patient is disease−free. A result labeled "No" or "False".

This is a binary classification problem. We have a set of observations called the training data set, which comprises sample data with actual classification results. We train a model, called Classifier on this data set, and use that model to predict whether a certain patient will have the disease or not.

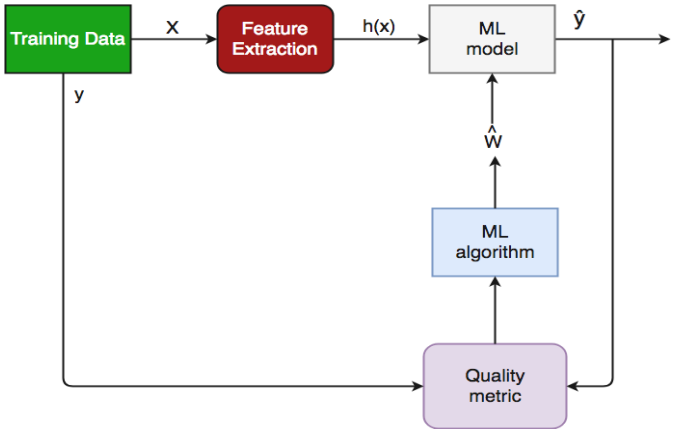
The outcome, thus now depends upon:

1. How well these features are able to "map" to the outcome.
2. The quality of our data set. By quality, I refer to statistical and Mathematical qualities.
3. How well our Classifier generalizes this relationship between the features and the outcome.
4. The values of the x1 and x2.

Following is the generalized block diagram of the classification task.

Generalized Classification Block Diagram:

1. X: pre−classified data, in the form of an N*M matrix. N is the no. of observations and M is the number of features
2. y: An N−d vector corresponding to predicted classes for each of the N observations.
3. Feature Extraction: Extracting valuable information from input X using a series of transforms.
4. ML Model: The "Classifier" we'll train.
5. y': Labels predicted by the Classifier.
6. Quality Metric: Metric used for measuring the performance of the model.
7. ML Algorithm: The algorithm that is used to update weights w', which updates the model and "learns" iteratively.



Types of Classifiers (algorithms)

There are various types of classifiers. Some of them are:

- Linear Classifiers: Logistic Regression
- Tree−Based Classifiers: Decision Tree Classifier
- Support Vector Machines
- Artificial Neural Networks
- Bayesian Regression
- Gaussian Naive Bayes Classifiers
- Stochastic Gradient Descent (SGD) Classifier
- Ensemble Methods: Random Forests, AdaBoost, Bagging Classifier, Voting Classifier, ExtraTrees Classifier

Learners in Classification Problems:

In the classification problems, there are two types of learners:

1. **Lazy Learners:** Lazy Learner firstly stores the training dataset and wait until it receives the test dataset. In Lazy learner case, classification is done on the basis of the most related data stored in the training dataset. It takes less time in training but more time for predictions. **Example:** K–NN algorithm, Case–based reasoning
2. **Eager Learners:** Eager Learners develop a classification model based on a training dataset before receiving a test dataset. Opposite to Lazy learners, Eager Learner takes more time in learning, and less time in prediction. **Example:** Decision Trees, Naïve Bayes, ANN.

Evaluating a Classification model:

Once our model is completed, it is necessary to evaluate its performance; either it is a Classification or Regression model. So, for evaluating a Classification model, we have the following ways:

1. **Log Loss or Cross-Entropy Loss:**
   - It is used for evaluating the performance of a classifier, whose output is a probability value between the 0 and 1.
   - For a good binary Classification model, the value of log loss should be near to 0.
   - The value of log loss increases if the predicted value deviates from the actual value.
   - The lower log loss represents the higher accuracy of the model.
   - For Binary classification, cross–entropy can be calculated as:

     1.? (ylog(p)+(1?y)log(1?p))

     Where y= Actual output, p= predicted output.
2. **Confusion Matrix:**
   - The confusion matrix provides us a matrix/table as output and describes the performance of the model. It is also known as the error matrix.
   - The matrix consists of predictions result in a summarized form, which has a total number of correct predictions and incorrect predictions. The matrix looks like as below table:

	Actual Positive	**Actual Negative**
Predicted Positive	True Positive	False Positive
Predicted Negative	False Negative	True Negative

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{Total Population}}$$

3. **AUC-ROC curve:**
   - ROC curve stands for **Receiver Operating Characteristics Curve** and AUC stands for **Area Under the Curve**.

- It is a graph that shows the performance of the classification model at different thresholds.
- To visualize the performance of the multi−class classification model, we use the AUC− ROC Curve.
- The ROC curve is plotted with TPR and FPR, where TPR (True Positive Rate) on Y−axis and FPR (False Positive Rate) on X−axis.

Use cases of Classification Algorithms

Classification algorithms can be used in different places. Below are some popular use cases of Classification Algorithms:

→ Email Spam Detection

→ Speech Recognition

→ Identifications of Cancer tumor cells.

→ Drugs Classification

→ Biometric Identification, etc.

III. REGRESSION

→ Regression in machine learning refers to a **supervised learning** technique where the goal is to predict a continuous numerical value based on one or more independent features. It finds relationships between variables so that predictions can be made. we have two types of variables present in regression:

- **Dependent Variable (Target)**: The variable we are trying to predict e.g house price.
- **Independent Variables (Features)**: The input variables that influence the prediction e.g locality, number of rooms.

Regression analysis problem works with if output variable is a real or continuous value such as “salary” or “weight”. Many different regression models can be used but the simplest model in them is linear regression.

Types of Regression

Regression can be classified into different types based on the number of predictor variables and the nature of the relationship between variables:

1. Simple Linear Regression

- **Linear regression** is one of the simplest and most widely used statistical models. This assumes that there is a linear relationship between the independent and dependent variables.
- This means that the change in the dependent variable is proportional to the change in the independent variables. For example, predicting the price of a house based on its size.

2. Multiple Linear Regression

- Multiple linear regression extends simple linear regression by using multiple independent variables to predict target variable. For example, predicting the price of a house based on multiple features such as size, location, number of rooms, etc.

3. Polynomial Regression

- Polynomial regression is used to model with non-linear relationships between the dependent variable and the independent variables.
- It adds polynomial terms to the linear regression model to capture more complex relationships. For example, when we want to predict a non-linear trend like population growth over time, we use polynomial regression.

4. **Ridge & Lasso Regression**

- Ridge & lasso regression are regularized versions of linear regression that help avoid overfitting by penalizing large coefficients. When there's a risk of overfitting due to too many features we use these type of regression algorithms.

5. Support Vector Regression (SVR)

- SVR is a type of regression algorithm that is based on the Support Vector Machine (SVM) algorithm.
- SVM is a type of algorithm that is used for classification tasks but it can also be used for regression tasks.
- SVR works by finding a hyperplane that minimizes the sum of the squared residuals between the predicted and actual values.

6. Decision Tree Regression

- Decision tree Uses a tree-like structure to make decisions where each branch of tree represents a decision and leaves represent outcomes.
- For example, predicting customer behaviour based on features like age, income, etc there we use decision tree regression.

7. Random Forest Regression

- Random Forest is an ensemble method that builds multiple decision trees and each tree is trained on a different subset of the training data. The final prediction is made by averaging the predictions of all of the trees. For example, customer churn or sales data using this.

Regression Evaluation Metrics:

Evaluation in machine learning measures the performance of a model. Here are some popular evaluation metrics for regression:

- **Mean Absolute Error (MAE):** The average absolute difference between the predicted and actual values of the target variable.
- **Mean Squared Error (MSE):** The average squared difference between the predicted and actual values of the target variable.
- **Root Mean Squared Error (RMSE):** Square root of the mean squared error.
- **Huber Loss:** A hybrid loss function that transitions from MAE to MSE for larger errors, providing balance between robustness and MSE's sensitivity to outliers.

- **R2 – Score:** Higher values indicate better fit ranging from 0 to 1.

Applications of Regression

→ **Predicting prices:** Used to predict the price of a house based on its size, location and other features.

→ **Forecasting trends:** Model to forecast the sales of a product based on historical sales data.

→ **Identifying risk factors:** Used to identify risk factors for heart patient based on patient medical data.

→ **Making decisions:** It could be used to recommend which stock to buy based on market data.

Advantages of Regression

→ Easy to understand and interpret.

→ Robust to outliers.

→ Can handle both linear relationships easily.

Disadvantages of Regression

→ Assumes linearity.

→ Sensitive to situation where two or more independent variables are highly correlated with each other i.e multicollinearity.

→ May not be suitable for highly complex relationships.

IV. APPROACHES OF MACHINE LEARNING MODELS

→ In today's world, Machine learning becomes one of the popular and exciting fields of study that gives machines the ability to learn and become more accurate at predicting outcomes for the unseen data i.e, not seen the data in prior.

→ The ideas in Machine learning overlaps and receives from **Artificial Intelligence** and many other related technologies. Today, machine learning is evolved from **Pattern Recognition** and the concept that computers can learn without being explicitly programmed to performing specific tasks.

→ We can use the Machine Learning algorithms (e.g, **Logistic Regression**, **Naive Bayes**, etc) to

- Recognize spoken words,
- Data Mining, and
- Build applications that learn from data, etc.

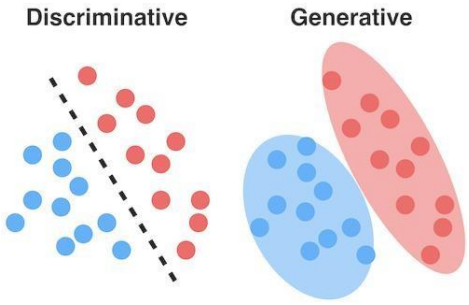
→ And the improvement of these algorithms in terms of accuracy increases over time.

→ Machine learning models can be classified into two types of models **Discriminative** and **Generative** models.

→ In simple words, a discriminative model makes predictions on the unseen data based on conditional probability and can be used either for classification or regression problem statements.

→ On the contrary, a generative model focuses on the distribution of a dataset to return a probability for a given example.

We as a human can adopt any of the two different approaches to machine learning models while learning an artificial language. These two models have not previously been explored in human learning. However, it is related to known effects of causal direction, classification vs. inference learning, and observational vs. feedback learning.



Problem Formulation

Suppose we are working on a classification problem where our task is to decide if an email is a spam or not spam based on the words present in a particular email. To solve this problem, we have a joint model over

- Labels: **Y=y**, and
- Features: **X = {x1, x2, …xn}**

Therefore, the joint distribution of the model can be represented as

$p(Y, X) = P(y, x1, x2, \ldots xn)$

Now, our goal is to estimate the probability of spam email i.e, **P(Y=1|X)**. Both generative and discriminative models can solve this problem but in different ways.

Let's see why and how they are different!

The approach of Generative Models

In the case of generative models, to find the conditional probability **P(Y|X)**, they estimate the prior probability **P(Y)** and likelihood probability **P(X|Y)** with the help of the training data and uses the Bayes Theorem to calculate the posterior probability **P(Y |X):**

$$posterior = \frac{prior \times likelihood}{evidence} \Rightarrow P(Y|X) = \frac{P(Y) \cdot P(X|Y)}{P(X)}$$

The approach of Discriminative Models

In the case of discriminative models, to find the probability, they directly assume some functional form for **P(Y|X)** and then estimate the parameters of **P(Y|X)** with the help of the training data.

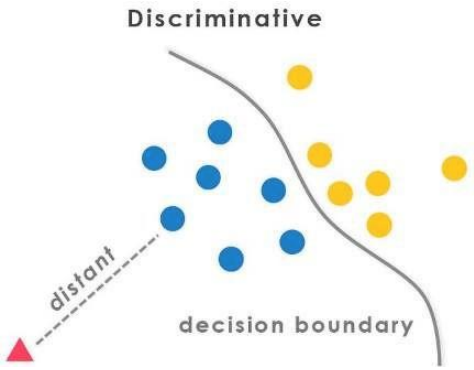
What are Discriminative Models?

→ The discriminative model refers to a class of models used in **Statistical Classification**, mainly used for supervised machine learning. These types of models are also known as **conditional models** since they learn the boundaries between classes or labels in a dataset.

→ Discriminative models (just as in the literal meaning) separate classes instead of modeling the

conditional probability and don't make any assumptions about the data points. But these models are not capable of generating new data points. Therefore, the ultimate objective of discriminative models is to separate one class from another.

→ If we have some outliers present in the dataset, then discriminative models work better compared to generative models i.e., discriminative models are more robust to outliers. However, there is one major drawback of these models is the **misclassification problem**, i.e., wrongly classifying a data point.



Training discriminative classifiers involve estimating a function **f: X -> Y**, or probability **P(Y|X)**

- Assume some functional form for the probability such as **P(Y|X)**
- With the help of training data, we estimate the parameters of **P(Y|X)**

Some Examples of Discriminative Models

- Logistic regression
- Scalar Vector Machine (SVMs)
- Traditional neural networks
- Nearest neighbor
- Conditional Random Fields (CRFs)
- Decision Trees and Random Forest

What are Generative Models?

Generative models are considered as a class of statistical models that can generate new data instances. These models are used in unsupervised machine learning as a means to perform tasks such as

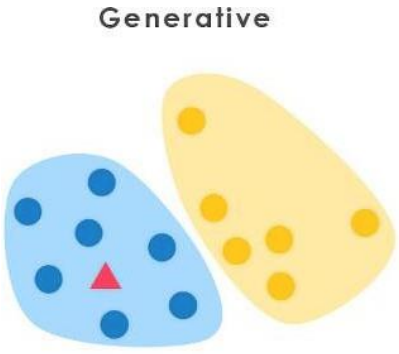
- Probability and Likelihood estimation,
- Modeling data points,
- To describe the phenomenon in data,
- To distinguish between classes based on these probabilities.

Since these types of models often rely on the Bayes theorem to find the joint probability, so generative models can tackle a more complex task than analogous discriminative models.

So, Generative models focus on the distribution of individual classes in a dataset and the learning

algorithms tend to model the underlying patterns or distribution of the data points. These models use the concept of joint probability and create the instances where a given **feature (x)** or input and the desired output or **label (y)** exist at the same time.

These models use **probability estimates** and **likelihood** to model data points and differentiate between different class labels present in a dataset. Unlike discriminative models, these models are also capable of generating new data points.



Mathematical things involved in Generative Models

Training generative classifiers involve estimating a function **f: X -> Y**, or probability **P(Y|X):**

- Assume some functional form for the probabilities such as **P(Y), P(X|Y)**
- With the help of training data, we estimate the parameters of **P(X|Y), P(Y)**
- Use the Bayes theorem to calculate the posterior probability **P(Y |X)**

Some Examples of Generative Models

- Naïve Bayes
- Bayesian networks
- Markov random fields
- Hidden Markov Models (HMMs)
- Latent Dirichlet Allocation (LDA)
- Generative Adversarial Networks (GANs)
- Autoregressive Model

Difference between Discriminative and Generative Models

Let's see some of the differences between Discriminative and Generative Models.

Core Idea

Discriminative models draw boundaries in the data space, while generative models try to model how data is placed throughout the space. A generative model focuses on explaining how the data was generated, while a discriminative model focuses on predicting the labels of the data.

Mathematical Intuition

In mathematical terms, a discriminative machine learning trains a model which is done by learning parameters that maximize the conditional probability **P(Y|X)**, while on the other hand, a generative model

learns parameters by maximizing the joint probability of **P(X, Y)**.

Applications

Discriminative models recognize existing data i.e, discriminative modeling identifies tags and sorts data and can be used to classify data while Generative modeling produces something.

Since these models use different approaches to machine learning, so both are suited for specific tasks i.e, Generative models are useful for unsupervised learning tasks while discriminative models are useful for supervised learning tasks.

Outliers

Generative models have more impact on outliers than discriminative models.

Computational Cost

Discriminative models are computationally cheap as compared to generative models.

Comparison between Discriminative and Generative Models

Let's see some of the comparisons based on the following criteria between Discriminative and Generative Models:

- Performance
- Missing Data
- Accuracy Score
- Applications

Based on Performance

Generative models need fewer data to train compared with discriminative models since generative models are more biased as they make stronger assumptions i.e, **assumption of conditional independence**.

Based on Missing Data

In general, if we have missing data in our dataset, then Generative models can work with these missing data, while on the contrary discriminative models can't. This is because, in generative models, still we can estimate the posterior by marginalizing over the unseen variables. However, for discriminative models, we usually require all the features X to be observed.

Based on Accuracy Score: If the assumption of conditional independence violates, then at that time generative models are less accurate than discriminative models.

Based on Applications: Discriminative models are called **"discriminative"** since they are useful for discriminating Y's label i.e, target outcome, so they can only solve classification problems while Generative models have more applications besides classification such as,

- Samplings,
- Bayes learning,
- MAP inference, etc.

V. TYPES OF LEARNING

The main types of learning in machine learning are categorized based on how the model learns from data and the nature of the data itself. These include:

- **Supervised Learning:**
  - **Description:** The model learns from labeled data, where each input example is paired with its corresponding correct output. The goal is to learn a mapping from inputs to outputs so that the model can predict outputs for new, unseen inputs.
  - **Examples:** Classification (predicting categories, e.g., spam detection) and Regression (predicting continuous values, e.g., house price prediction).
- **Unsupervised Learning:**
  - **Description:** The model learns from unlabeled data, aiming to discover hidden patterns, structures, or relationships within the data without explicit guidance.
  - **Examples:** Clustering (grouping similar data points, e.g., customer segmentation) and Dimensionality Reduction (reducing the number of features while retaining important information).
- **Reinforcement Learning:**
  - **Description:** An agent learns to make decisions by interacting with an environment. It receives rewards for desirable actions and penalties for undesirable ones, aiming to maximize cumulative reward over time.
  - **Examples:** Game playing (e.g., AlphaGo) and Robotics (teaching robots to perform tasks).
- **Semi-supervised Learning:**
  - **Description:** This approach combines aspects of both supervised and unsupervised learning. It utilizes a small amount of labeled data along with a larger amount of unlabeled data to train a model.
  - **Examples:** Text classification with limited labeled documents, image recognition.
- **Self-supervised Learning:**
  - **Description:** A subset of unsupervised learning where the model generates its own labels from the input data itself, effectively creating a supervised learning task from unlabeled data.
  - **Examples:** Pre-training large language models (like BERT or GPT) by predicting masked words or next sentences.

5. Deep Learning:

- **Concept:**

  A subfield of machine learning that utilizes artificial neural networks with multiple layers (deep neural networks) to learn complex patterns from large datasets. Deep learning models can be applied to supervised, unsupervised, and reinforcement learning tasks.

- **Examples:**
  - **Convolutional Neural Networks (CNNs):** Primarily used for image and video analysis.
  - **Recurrent Neural Networks (RNNs) and Transformers:** Primarily used for sequential data like natural language processing (NLP) and time series.

VI. PROBABILITY - BASICS

Probability in machine learning:

→ Probability is the bedrock of ML, which tells how likely is the event to occur. The value of Probability always lies between 0 to 1. It is the core concept as well as a primary prerequisite to understanding the ML models and their applications.

→ **Probability can be calculated by the number of times the event occurs divided by the total number of possible outcomes**. Let's suppose we tossed a coin, then the probability of getting head as a possible outcome can be calculated as below formula:

P (H) = Number of ways to head occur/ total number of possible outcomes

P (H) = ½

P (H) = 0.5

Where,

P (H) = Probability of occurring Head as outcome while tossing a coin.

Types of Probability

For better understanding the Probability, it can be categorized further in different types as follows:

- **Empirical Probability:** Empirical Probability can be calculated as the number of times the event occurs divided by the total number of incidents observed.
- **Theoretical Probability:** Theoretical Probability can be calculated as the number of ways the particular event can occur divided by the total number of possible outcomes.
- **Joint Probability:** It tells the Probability of simultaneously occurring two random events.

$P(A \cap B) = P(A).\ P(B)$

Where,

$P(A \cap B)$ = Probability of occurring events A and B both.

P (A) = Probability of event A

P (B) = Probability of event B

- **Conditional Probability:** It is given by the Probability of event A given that event B occurred.

The Probability of an event A conditioned on an event B is denoted and defined as;

$P(A|B) = P(A \cap B)/P(B)$

Similarly, $P(B|A) = P(A \cap B)/ P(A)$. We can write the joint Probability of as A and B as $P(A \cap B)= p(A).P(B|A)$, which means: "The chance of both things happening is the chance that the first one happens, and then the second one is given when the first thing happened."

We have a basic understanding of Probability required to learn Machine Learning.

Now, we will discuss the basic introduction of Statistics for ML.

Statistics in Machine Learning:

Statistics is also considered as the base foundation of machine learning which deals with finding answers to the questions that we have about data. In general, we can define statistics as:

Statistics can be categorized into 2 major parts. These are as follows:

→ Descriptive Statistics

→ Inferential Statistics

Use of Statistics in ML:

Statistics methods are used to understand the training data as well as interpret the results of testing different machine learning models. Further, Statistics can be used to make better−informed business and investing decisions.

Conditional probability and Bayesian theorem:

Conditional probabilities arise naturally in the investigation of experiments where an outcome of a trial may affect the outcomes of the subsequent trials. We try to calculate the probability of the second event (event B) given that the first event (event A) has already happened. If the probability of the event changes when we take the first event into consideration, we can safely say that the probability of event B is dependent of the occurrence of event A.

Let's think of cases where this happens:

→ Drawing a second ace from a deck given we got the first ace

→ Finding the probability of having a disease given you were tested positive

→ Finding the probability of liking Harry Potter given we know the person likes fiction

And so on….

Here we can define, 2 events:

→ Event A is the probability of the event we're trying to calculate.

→ Event B is the condition that we know or the event that has happened.

We can write the conditional probability as, $P(\frac{A}{B})$, the probability of the occurrence of event A given that B has already happened.

$$P\left(\frac{A}{B}\right) = \frac{P(\text{A and B})}{P(B)} = \frac{\text{Probability of the occurence of both A and B}}{\text{Probability of B}}$$

Bayes Theorem

→ **Bayesian decision theory** refers to the statistical approach based on trade-off quantification among various classification decisions based on the concept of Probability(Bayes Theorem) and the costs associated with the decision.

→ It is basically a classification technique that involves the use of the Bayes Theorem which is used to find the conditional probabilities.

→ The Bayes theorem describes the probability of an event based on the prior knowledge of the conditions that might be related to the event. The conditional probability of A given B, represented by P(A | B) is the chance of occurrence of A given that B has occurred.

$$P(A \mid B) = P(A,B)/P(B) \text{ or}$$

By Using the Chain rule, this can also be written as:

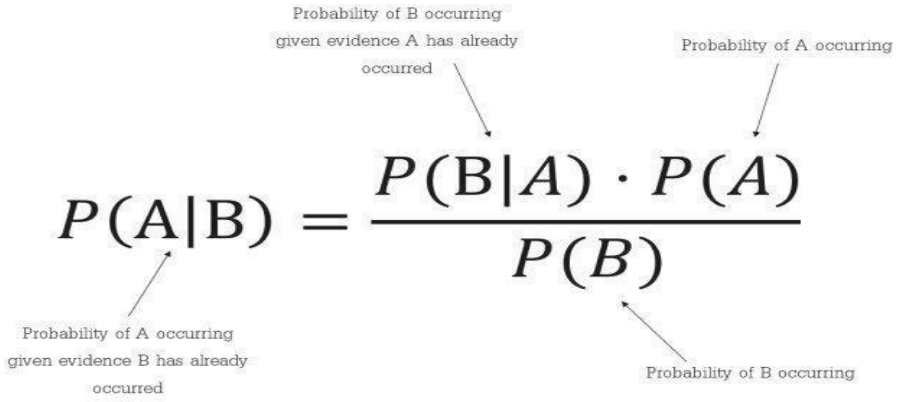
$$P(A, B) = P(A|B)P(B)=P(B|A)P(A)$$

$$P(A \mid B) = P(B|A)P(A)/P(B) \quad \text{—— (1)}$$

Where,

$$P(B) = P(B,A) + P(B,A') = P(B|A)P(A) + P(B|A')P(A')$$

Here, equation (1) is known as the **Bayes Theorem of probability**



Our aim is to explore each of the components included in this theorem. Let's explore step by step:

a) Prior or State of Nature:

→ Prior probabilities represent how likely is each Class is going to occur.

→ Priors are known before the training process.

→ The state of nature is a random variable **P(wi)**.

→ If there are only two classes, then the sum of the priors is **P(w1) + P(w2)=1**, if the classes are exhaustive.

b) Class Conditional Probabilities:

→ It represents the probability of how likely a feature x occurs given that it belongs to the particular class. It is denoted by, **P(X|A)** where x is a particular feature

→ It is the probability of how likely the feature x occurs given that it belongs to the class wi.

→ Sometimes, it is also known as the **Likelihood**.

→ It is the quantity that we have to evaluate while training the data. During the training process, we have input(features) X labeled to corresponding class w and we figure out the likelihood of occurrence of that set of features given the class label.

Evidence:

→ It is the probability of occurrence of a particular feature i.e. **P(X)**.

→ It can be calculated using the chain rule as, **P(X) = Σin P(X | wi) P(wi).**

→ As we need the likelihood of class conditional probability is also figure out evidence values during training.

Posterior Probabilities:

→ It is the probability of occurrence of Class A when certain Features are given

→ It is what we aim at computing in the test phase in which we have testing input/features (the given entity) & have to find how likely trained model can predict features belonging to the particular class wi.

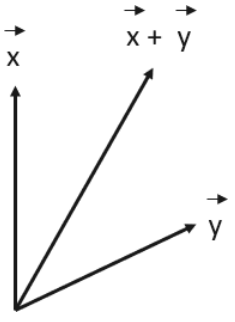
VII. LINEAR ALGEBRA

What is Linear Algebra?

This is a branch of mathematic that concerns the study of the vectors and certain rules to manipulate the vector. When we are formalizing intuitive concepts, the common approach is to construct a set of objects (symbols) and a set of rules to manipulate these objects. This is what we knew as algebra.

If we talk about Linear Algebra in machine learning, it is defined as the part of mathematics that uses vector space and matrices to represent **linear equations.**

When talking about vectors, people might flashback to their high school study regarding the vector with direction, just like the image below.



Geometric Vector

This is a vector, but not the kind of vector discussed in the Linear Algebra for Machine Learning. Instead, it would be this image below we would talk about.

$$y = \begin{bmatrix} 10 \\ 20 \\ 56 \\ 77 \end{bmatrix} \in \mathbb{R}^4$$

vector 4x1 Matrix

What we had above is also a vector, but another kind of vector. You might be familiar with matrix form (the image below). The vector is a matrix with only 1 column, which is known as a column vector. In other words, we can think of a matrix as a group of column vectors or row vectors. In summary, vectors are special objects that can be added together and multiplied by scalars to produce another object of the same kind. We could have various objects called vectors.

$$y = \begin{bmatrix} 10 & 40 \\ 20 & 30 \\ 56 & 84 \\ 77 & 96 \end{bmatrix} \in \mathbb{R}^{4x2}$$

Matrix

→ Linear algebra itself s a systematic representation of data that computers can understand, and all the operations in linear algebra are systematic rules. That is why in modern time machine learning, Linear algebra is an important study.

→ An example of how linear algebra is used is in the linear equation. Linear algebra is a tool used in the Linear Equation because so many problems could be presented systematically in a Linear way. The typical Linear equation is presented in the form below.

$$a_{m1}x_1 + \ldots + a_{mn}x_n = b_m$$

Linear Equation

To solve the linear equation problem above, we use Linear Algebra to present the linear equation in a systematical representation. This way, we could use the matrix characterization to look for the most optimal solution.

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

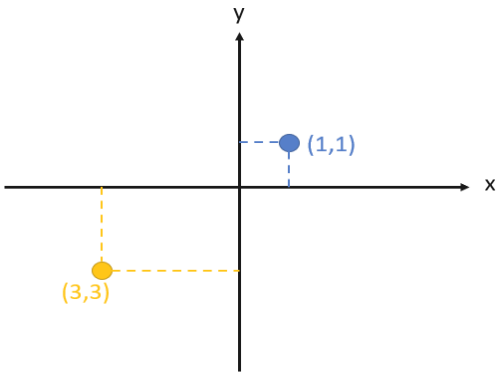
Linear Equation in Matrix Representation

To summary the Linear Algebra subject, there are three terms you might want to learn more as a starting point within this subject:

- Vector
- Matrix
- Linear Equation

Analytic Geometry (Coordinate Geometry)

Analytic geometry is a study in which we learn the data (point) position using an ordered pair of coordinates. This study is concerned with defining and representing geometrical shapes numerically and extracting numerical information from the shapes numerical definitions and representations. We project the data into the plane in a simpler term, and we receive numerical information from there.



Cartesian Coordinate

Above is an example of how we acquired information from the data point by projecting the dataset into the plane. How we acquire the information from this representation is the heart of Analytical Geometry. To help you start learning this subject, here are some important terms you might need.

Distance Function

A **distance function** is a function that provides numerical information for the distance between the elements of a set. If the distance is zero, then elements are equivalent. Else, they are different from each other.

An example of the distance function is Euclidean Distance which calculates the linear distance between two data points.

$$\sqrt{(q_2 - q_1)^2 + (p_2 - p_1)^2}$$

Euclidean Distance Equation

Inner Product

The inner product is a concept that introduces intuitive geometrical concepts, such as the **length of a vector** and the **angle or distance between two vectors**. It is often denoted as

⟨x,y⟩ (or occasionally (x,y) or ⟨x|y⟩).

VIII. HYPOTHESIS SPACE.

Hypothesis Space

→ The hypothesis space (H) in machine learning refers to the set of all possible functions or models that a learning algorithm can potentially consider to explain the relationship between input features and target outputs.

→ The concept of a hypothesis is fundamental in Machine Learning and data science endeavours. In the realm of machine learning, a hypothesis serves as an initial assumption made by data scientists and ML professionals when attempting to address a problem. Machine learning involves conducting experiments based on past experiences, and these hypotheses are crucial in formulating potential solutions.

Hypothesis in Machine Learning

A hypothesis in machine learning is the model's presumption regarding the connection between the input features and the result. It is an illustration of the mapping function that the algorithm is attempting to discover using the training set. To minimize the discrepancy between the expected and actual outputs, the learning process involves modifying the weights that parameterize the hypothesis. The objective is to optimize the model's parameters to achieve the best predictive performance on new, unseen data, and a cost function is used to assess the hypothesis' accuracy.

How does a Hypothesis work?

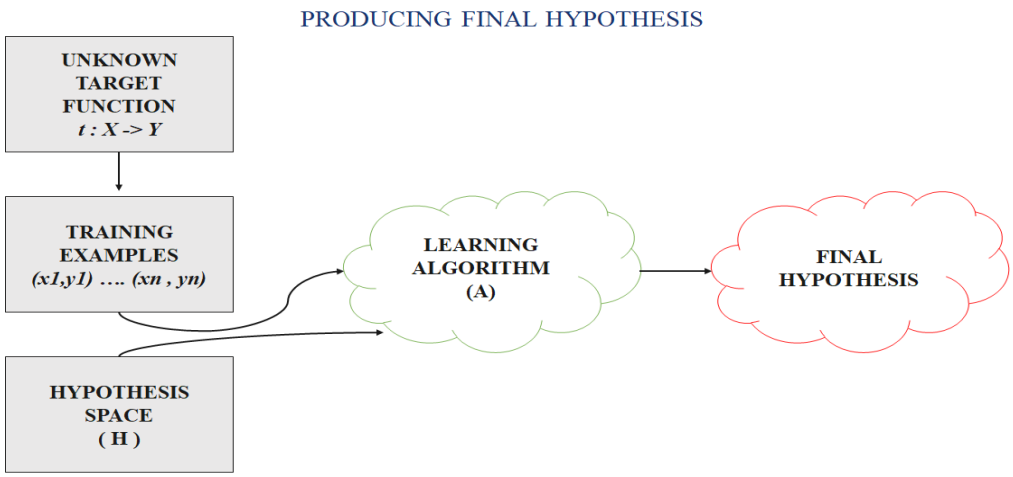
In most supervised machine learning algorithms, our main goal is to find a possible hypothesis from the hypothesis space that could map out the inputs to the proper outputs. The following figure shows the common method to find out the possible hypothesis from the Hypothesis space:

Hypothesis Space (H)

Hypothesis space is the set of all the possible legal hypothesis. This is the set from which the machine learning algorithm would determine the best possible (only one) which would best describe the target function or the outputs.

Hypothesis (h)

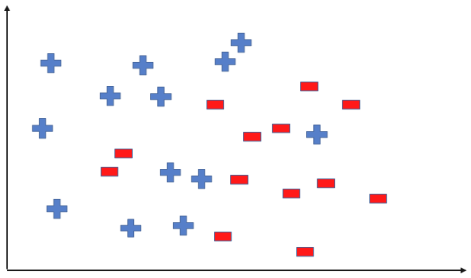
A hypothesis is a function that best describes the target in supervised machine learning. The hypothesis that an algorithm would come up depends upon the data and also depends upon the restrictions and bias that we have imposed on the data.



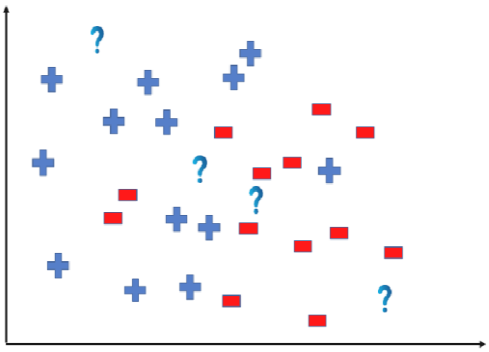
The Hypothesis can be calculated as:

y=mx+b → where, y = range, m = slope of the lines, x = domain, b = intercept

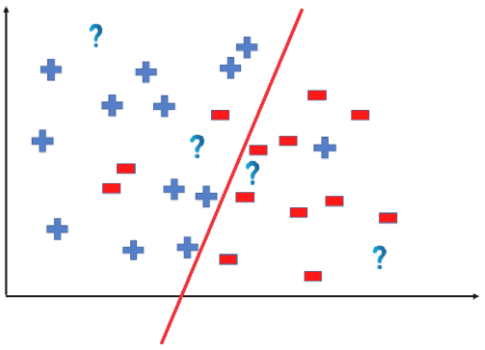
To better understand the Hypothesis Space and Hypothesis consider the following coordinate that shows the distribution of some data:



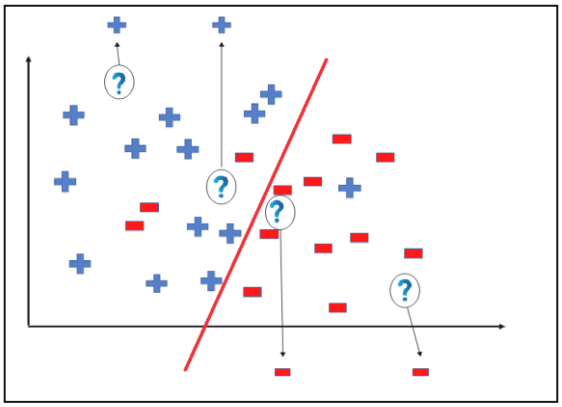
Say suppose we have test data for which we have to determine the outputs or results. The test data is as shown below:



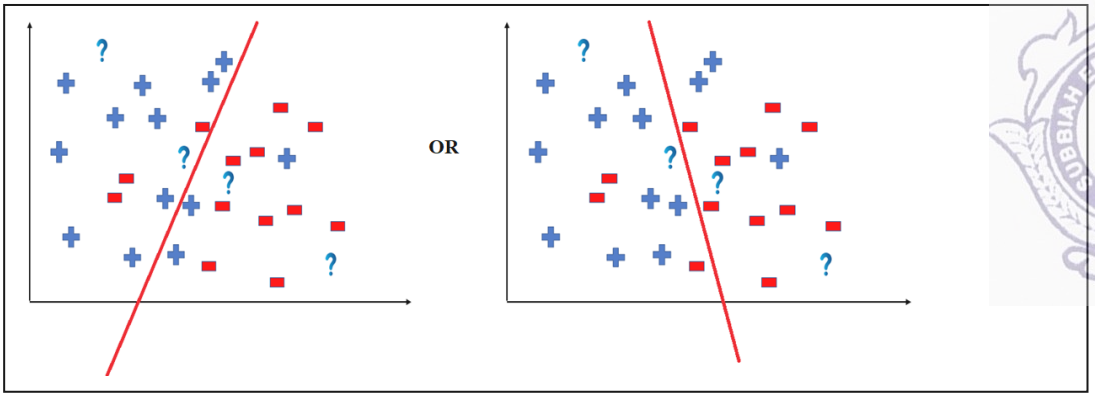
We can predict the outcomes by dividing the coordinate as shown below:



So the test data would yield the following result:



But note here that we could have divided the coordinate plane as:

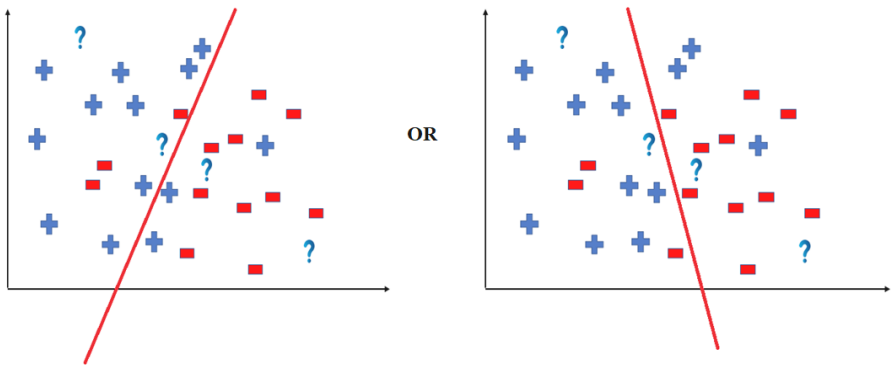


→ The way in which the coordinate would be divided depends on the data, algorithm and constraints.

→ All these legal possible ways in which we can divide the coordinate plane to predict the outcome of the test data composes of the Hypothesis Space.

→ Each individual possible way is known as the hypothesis.

→ Hence, in this example the hypothesis space would be like:



Hypothesis Space and Representation in Machine Learning

The hypothesis space comprises all possible legal hypotheses that a machine learning algorithm can consider. Hypotheses are formulated based on various algorithms and techniques, including linear regression, decision trees, and neural networks. These hypotheses capture the mapping function transforming input data into predictions.

Hypothesis Formulation and Representation in Machine Learning

Hypotheses in machine learning are formulated based on various algorithms and techniques, each with its representation. For example:

Linear Regression: h(X)=θ0+θ1X1+θ2X2+...+θnXnh(X)=θ0+θ1X1+θ2X2+...+θnXn

Decision Trees: h(X)=Tree(X)h(X)=Tree(X)

Neural Networks: h(X)=NN(X)h(X)=NN(X)

In the case of complex models like neural networks, the hypothesis may involve multiple layers of interconnected nodes, each performing a specific computation.

Hypothesis Evaluation:

The process of machine learning involves not only formulating hypotheses but also evaluating their performance. This evaluation is typically done using a loss function or an evaluation metric that quantifies the disparity between predicted outputs and ground truth labels. Common evaluation metrics include mean squared error (MSE), accuracy, precision, recall, F1-score, and others. By comparing the predictions of the hypothesis with the actual outcomes on a validation or test dataset, one can assess the effectiveness of the model.

Hypothesis Testing and Generalization:

Once a hypothesis is formulated and evaluated, the next step is to test its generalization capabilities. Generalization refers to the ability of a model to make accurate predictions on unseen data. A hypothesis that performs well on the training dataset but fails to generalize to new instances is said to suffer from overfitting. Conversely, a hypothesis that generalizes well to unseen data is deemed robust and reliable.

The process of hypothesis formulation, evaluation, testing, and generalization is often iterative in nature. It involves refining the hypothesis based on insights gained from model performance, feature importance, and domain knowledge. Techniques such as hyperparameter tuning, feature engineering, and model selection play a crucial role in this iterative refinement process.

Hypothesis in Statistics

In statistics, a hypothesis refers to a statement or assumption about a population parameter. It is a proposition or educated guess that helps guide statistical analyses. There are two types of hypotheses: the null hypothesis (H0) and the alternative hypothesis (H1 or Ha).

Null Hypothesis(H0): This hypothesis suggests that there is no significant difference or effect, and any observed results are due to chance. It often represents the status quo or a baseline assumption.

Aternative Hypothesis(H1 or Ha): This hypothesis contradicts the null hypothesis, proposing that there is a significant difference or effect in the population. It is what researchers aim to support with evidence.

IX. INDUCTIVE BIAS

What is Inductive Bias in Machine Learning?

→ In the realm of machine learning, the concept of inductive bias plays a pivotal role in shaping how algorithms learn from data and make predictions.

→ It serves as a guiding principle that helps algorithms generalize from the training data to unseen data, ultimately influencing their performance and decision-making processes. In this article, we delve into the intricacies of inductive bias, its significance in machine learning, and its implications for model development and interpretation.

What is Inductive Bias?

→ Inductive bias can be defined as the set of assumptions or biases that a learning algorithm employs to make predictions on unseen data based on its training data. These assumptions are inherent in the algorithm's design and serve as a foundation for learning and generalization.

→ The inductive bias of an algorithm influences how it selects a hypothesis (a possible explanation or model) from the hypothesis space (the set of all possible hypotheses) that best fits the training data. It helps the algorithm navigate the trade-off between fitting the training data perfectly (overfitting) and generalizing well to unseen data (underfitting).

Types of Inductive Bias

Inductive bias can manifest in various forms, depending on the algorithm and its underlying assumptions. Some common types of inductive bias include:

- **Bias towards simpler explanations**: Many machine learning algorithms, such as decision trees and linear models, have a bias towards simpler hypotheses. They prefer explanations that are more parsimonious and less complex, as these are often more likely to generalize well to unseen data.
- **Bias towards smoother functions**: Algorithms like kernel methods or Gaussian processes have a bias towards smoother functions. They assume that neighbouring points in the input space should have similar outputs, leading to smooth decision boundaries.
- **Bias towards specific types of functions**: Neural networks, for example, have a bias towards learning complex, nonlinear functions. This bias allows them to capture intricate patterns in the data but can also lead to overfitting if not regularized properly.
- **Bias towards sparsity**: Some algorithms, like Lasso regression, have a bias towards sparsity. They prefer solutions where only a few features are relevant, which can improve interpretability and generalization.

Importance of Inductive Bias

→ Inductive bias is crucial in machine learning as it helps algorithms generalize from limited training data to unseen data. Without a well-defined inductive bias, algorithms may struggle to make accurate predictions or may overfit the training data, leading to poor performance on new data.

→ Understanding the inductive bias of an algorithm is essential for model selection, as different biases may be more suitable for different types of data or tasks. It also provides insights into how the algorithm is learning and what assumptions it is making about the data, which can aid in interpreting its predictions and results.

Challenges and Considerations

→ While inductive bias is essential for learning, it can also introduce limitations and challenges. Biases that are too strong or inappropriate for the data can lead to poor generalization or biased predictions. Balancing bias with variance (the variability of predictions) is a key challenge in machine learning, requiring careful tuning and model selection.

→ Additionally, the choice of inductive bias can impact the interpretability of the model. Simpler biases may lead to more interpretable models, while more complex biases may sacrifice interpretability for improved performance.

Inductive bias is a fundamental concept in machine learning that shapes how algorithms learn and generalize from data. It serves as a guiding principle that influences the selection of hypotheses and the generalization of models to unseen data. Understanding the inductive bias of an algorithm is essential for model development, selection, and interpretation, as it provides insights into how the algorithm is learning and making predictions. By carefully considering and balancing inductive bias, machine learning practitioners can develop models that generalize well and provide valuable insights into complex datasets.

X. EVALUATION

→ Evaluation in machine learning is the process of assessing the performance of a trained model or hypothesis. This is crucial for understanding how well the model generalizes to new, unseen data and for comparing different models.

→ Evaluation typically involves:

- Splitting Data: Dividing the available dataset into training, validation, and test sets. The model is trained on the training set, hyper-parameters are tuned using the validation set, and the final performance is measured on the unseen test set.
- Metrics: Using appropriate metrics to quantify performance.

Example: For classification, metrics like accuracy, precision, recall, F1-score, or AUC-ROC are used. For regression, metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or R-squared are common.

- Employing techniques like k-fold cross-validation to get a more robust estimate of the model's performance, especially when data is limited.

Example: After training a classification model, we can evaluate its performance by calculating its accuracy on a held-out test set. If the accuracy is 85%, it indicates that the model correctly classifies 85% of the examples in the test set.

XI. TRAINING AND TEST SETS, CROSS VALIDATION, CONCEPT OF OVER FITTING, UNDER FITTING, BIAS AND VARIANCE.

1. Training and Test Sets

Training Set:

This is the portion of the dataset used to train the machine learning model. The model learns patterns and relationships from this data.

Test Set:

This is the unseen portion of the dataset used to evaluate the model's performance on new, unobserved data. It assesses how well the model generalizes.

Example: Imagine training a model to predict house prices. The training set would contain features (size, location, number of rooms) and corresponding prices for houses the model learns from. The test set would contain similar features for houses the model hasn't seen, and its predictions would be compared to the actual prices to gauge accuracy.

2. Cross-Validation

Cross-validation is a technique to estimate a model's performance and robustness more reliably than a single train-test split. It involves partitioning the data into multiple subsets (folds).

K-Fold Cross-Validation: The dataset is divided into 'k' equal-sized folds. The model is trained 'k' times, each time using 'k-1' folds for training and one fold for testing. The results are then averaged.

Example: For a 5-fold cross-validation, the data is split into 5 parts. In the first iteration, folds 2-5 are used for training, and fold 1 for testing. In the second, folds 1, 3-5 train, and fold 2 tests, and so on.

3. Overfitting and Underfitting

Overfitting occurs when a model learns the training data too well, including its noise and outliers, leading to poor performance on new, unseen data.

Example: A model trained to classify images of cats and dogs overfits if it perfectly identifies the cats and dogs in the training set but fails to recognize new cat and dog images because it memorized specific features of the training images instead of learning general characteristics.

Underfitting occurs when a model is too simple to capture the underlying patterns in the training data, resulting in poor performance on both training and test data.

Example: Using a simple linear regression model to predict a non-linear relationship (e.g., a curved trend) between two variables would likely underfit, as the model cannot capture the complexity of the data.

→ Overfitting and Underfitting are the two main problems that occur in machine learning and degrade the performance of the machine learning models.

→ The main goal of each machine learning model is **to generalize well**. Here **generalization** defines the ability of an ML model to provide a suitable output by adapting the given set of unknown input. It means after providing training on the dataset, it can produce reliable and accurate output.

→ Hence, the underfitting and overfitting are the two terms that need to be checked for the performance of the model and whether the model is generalizing well or not.

Before understanding the overfitting and underfitting, let's understand some basic term that will help to understand this topic well:

Signal: It refers to the true underlying pattern of the data that helps the machine learning model to learn from the data.

Noise: Noise is unnecessary and irrelevant data that reduces the performance of the model.

Bias: Bias is a prediction error that is introduced in the model due to oversimplifying the machine learning algorithms. Or it is the difference between the predicted values and the actual values.

Variance: If the machine learning model performs well with the training dataset, but does not perform well with the test dataset, then variance occurs.

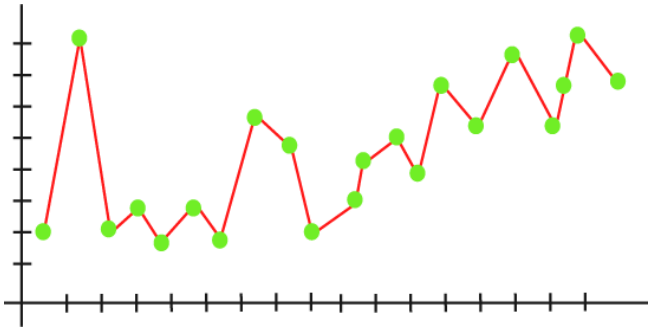
Overfitting

→ Overfitting occurs when our machine learning model tries to cover all the data points or more than the required data points present in the given dataset. Because of this, the model starts caching noise and inaccurate values present in the dataset, and all these factors reduce the efficiency and accuracy of the model. The overfitted model has **low bias** and **high variance.**

→ The chances of occurrence of overfitting increase as much we provide training to our model. It means the more we train our model, the more chances of occurring the overfitted model.

→ Overfitting is the main problem that occurs in supervised learning.

Example: The concept of the overfitting can be understood by the below graph of the linear regression output:



As we can see from the above graph, the model tries to cover all the data points present in the scatter plot. It may look efficient, but in reality, it is not so. Because the goal of the regression model to find the best fit line, but here we have not got any best fit, so, it will generate the prediction errors.

How to avoid the Overfitting in Model

Both overfitting and underfitting cause the degraded performance of the machine learning model. But the main cause is overfitting, so there are some ways by which we can reduce the occurrence of overfitting in our model.

→ Cross−Validation

→ Training with more data

→ Removing features

→ Early stopping the training

→ Regularization

→ Ensembling

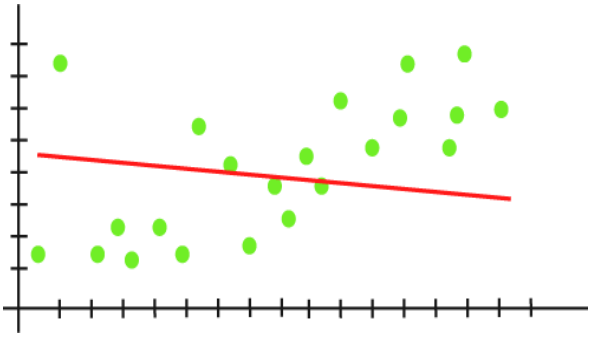
Underfitting

→ Underfitting occurs when our machine learning model is not able to capture the underlying trend of the data.

→ To avoid the overfitting in the model, the fed of training data can be stopped at an early stage, due to which the model may not learn enough from the training data. As a result, it may fail to find the best fit of the dominant trend in the data.

→ In the case of underfitting, the model is not able to learn enough from the training data, and hence it reduces the accuracy and produces unreliable predictions. An underfitted model has high bias and low variance.

Example: We can understand the underfitting using below output of the linear regression model:



As we can see from the above diagram, the model is unable to capture the data points present in the plot.

How to avoid underfitting:

→ By increasing the training time of the model.

→ By increasing the number of features.

Goodness of Fit

→ The "Goodness of fit" term is taken from the statistics, and the goal of the machine learning models to achieve the goodness of fit. In statistics modeling, it defines how closely the result or predicted values match the true values of the dataset.

→ The model with a good fit is between the underfitted and overfitted model, and ideally, it makes predictions with 0 errors, but in practice, it is difficult to achieve it.

→ As when we train our model for a time, the errors in the training data go down, and the same happens with test data. But if we train the model for a long duration, then the performance of the model may decrease due to the overfitting, as the model also learn the noise present in the dataset.

→ The errors in the test dataset start increasing, so the point, just before the raising of errors, is the good point, and we can stop here for achieving a good model. There are two other methods by which we can get a good point for our model, which are the **resampling method** to estimate model accuracy and **validation dataset**.

Cross validation:

→ Cross−validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. **We can also say that it is a technique to check how a statistical model generalizes to an independent dataset**.

→ In machine learning, there is always the need to test the stability of the model. It means based only on the training dataset; we can't fit our model on the training dataset. For this purpose, we reserve a particular sample of the dataset, which was not part of the training dataset. After that, we test our model on that sample before deployment, and this complete process comes under cross−validation. This is something different from the general train−test split.

→ Hence the basic steps of cross−validations are:

- Reserve a subset of the dataset as a validation set.
- Provide the training to the model using the training dataset.
- Now, evaluate model performance using the validation set. If the model performs well with the validation set, perform the further step, else check for the issues.

Methods used for Cross-Validation

There are some common methods that are used for cross−validation. These methods are given below:

→ Validation Set Approach

→ Leave−P−out cross−validation

→ Leave one out cross−validation

→ K−fold cross−validation

→ Stratified k−fold cross−validation

Validation Set Approach

→ We divide our input dataset into a training set and test or validation set in the validation set approach. Both the subsets are given 50% of the dataset.

→ But it has one of the big disadvantages that we are just using a 50% dataset to train our model, so the model may miss out to capture important information of the dataset. It also tends to give the underfitted model.

Leave-P-out cross-validation

→ In this approach, the p datasets are left out of the training data. It means, if there are total n datapoints in the original input dataset, then n−p data points will be used as the training dataset and the p data points as the validation set. This complete process is repeated for all the samples, and the average error is calculated to know the effectiveness of the model.

→ There is a disadvantage of this technique; that is, it can be computationally difficult for the large p.

Leave one out cross-validation

→ This method is similar to the leave−p−out cross−validation, but instead of p, we need to take 1 dataset out of training. It means, in this approach, for each learning set, only one datapoint is reserved, and the remaining dataset is used to train the model. This process repeats for each datapoint. Hence for n samples, we get n different training set and n test set.

→ It has the following features:

- In this approach, the bias is minimum as all the data points are used.
- The process is executed for n times; hence execution time is high.
- This approach leads to high variation in testing the effectiveness of the model as we iteratively check against one data point.

K-Fold Cross-Validation

K−fold cross−validation approach divides the input dataset into K groups of samples of equal sizes. These samples are called **folds**. For each learning set, the prediction function uses k−1 folds, and the rest of the folds are used for the test set. This approach is a very popular CV approach because it is easy to understand, and the output is less biased than other methods.

- The steps for k−fold cross−validation are:
- Split the input dataset into K groups
    - For each group:
    - Take one group as the reserve or test data set.
    - Use remaining groups as the training dataset
    - Fit the model on the training set and evaluate the performance of the model using the test set.

Stratified k-fold cross-validation

→ This technique is similar to k−fold cross−validation with some little changes. This approach works on stratification concept, it is a process of rearranging the data to ensure that each fold or group is a good representative of the complete dataset. To deal with the bias and variance, it is one of the best approaches.

→ It can be understood with an example of housing prices, such that the price of some houses can be much high than other houses. To tackle such situations, a stratified k−fold cross− validation technique is useful.

Holdout Method

→ This method is the simplest cross−validation technique among all. In this method, we need to remove a subset of the training data and use it to get prediction results by training it on the rest part of the dataset.

→ The error that occurs in this process tells how well our model will perform with the unknown dataset. Although this approach is simple to perform, it still faces the issue of high variance, and it also produces misleading results sometimes.

Comparison of Cross–validation to train/test split in Machine Learning

- **Train/test split:** The input data is divided into two parts, that are training set and test set on a ratio of 70:30, 80:20, etc. It provides a high variance, which is one of the biggest disadvantages.
- **Training Data:** The training data is used to train the model, and the dependent variable is known.
- **Test Data:** The test data is used to make the predictions from the model that is already trained on the training data. This has the same features as training data but not the part of that.

Cross-Validation dataset: It is used to overcome the disadvantage of train/test split by splitting the dataset into groups of train/test splits, and averaging the result. It can be used if we want to optimize our model that has been trained on the training dataset for the best performance. It is more efficient as compared to train/test split as every observation is used for the training and testing both.

Limitations of Cross-Validation

There are some limitations of the cross−validation technique, which are given below:

→ For the ideal conditions, it provides the optimum output. But for the inconsistent data, it may produce a drastic result. So, it is one of the big disadvantages of cross− validation, as there is no certainty of the type of data in machine learning.

→ In predictive modelling, the data evolves over a period, due to which, it may face the differences between the training set and validation sets. Such as if we create a model for the prediction of stock market values, and the data is trained on the previous 5 years stock values, but the realistic future values for the next 5 years may drastically different, so it is difficult to expect the correct output for such situations.

5. Bias and Variance

Bias:

The error introduced by approximating a real-world problem, which may be complex, by a simplified model. High bias leads to underfitting.

Example: Assuming a linear relationship when the data is clearly non-linear.

Variance:

The error introduced by the model's sensitivity to small fluctuations in the training data. High variance leads to overfitting.

Example: A very complex model that perfectly fits the training data, including its noise, will likely have high variance and perform poorly on new data.

Bias-Variance Trade-off: There is an inherent trade-off between bias and variance. Reducing bias often increases variance, and vice versa. The goal is to find a model that balances these two errors for optimal generalization performance.

XII. REGRESSION:

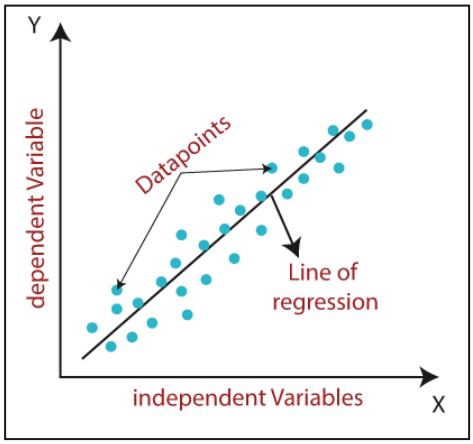
Regression modeling: Predicting continuous values**.**

1. LINEAR REGRESSION:

→ Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price,** etc.

→ Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:



Mathematically, we can represent a linear regression as:

$$y= a0+a1x+ \varepsilon$$

Here,

Y= Dependent Variable (Target Variable)

X= Independent Variable (predictor Variable)

a0= intercept of the line (Gives an additional degree of freedom)

a1 = Linear regression coefficient (scale factor to each input value).

ε = random error

The values for x and y variables are training datasets for Linear Regression model representation.

Types of Linear Regression

Linear regression can be further divided into two types of the algorithm:

- **Simple Linear Regression:**

  If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.

- **Multiple Linear regression:**

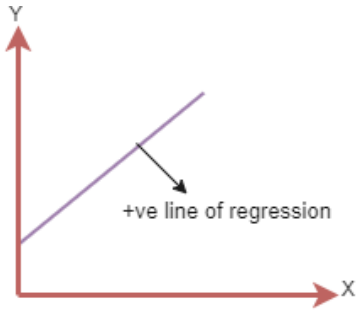
If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

Linear Regression Line

A linear line showing the relationship between the dependent and independent variables is called a **regression line**. A regression line can show two types of relationship:

- **Positive Linear Relationship:**

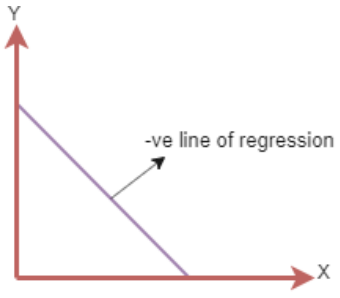
If the dependent variable increases on the Y−axis and independent variable increases on X−axis, then such a relationship is termed as a Positive linear relationship.



The line equation will be: $Y= a_0+a_1x$

- **Negative Linear Relationship:**

If the dependent variable decreases on the Y−axis and independent variable increases on the X−axis, then such a relationship is called a negative linear relationship.



The line of equation will be: $Y= -a_0+a_1x$

Finding the best fit line:

→ When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.

→ The different values for weights or the coefficient of lines (a0, a1) gives a different line of regression, so we need to calculate the best values for a0 and a1 to find the best fit line, so to calculate this we use cost function.

Cost function

→ The different values for weights or coefficient of lines (a0, a1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line.

→ Cost function optimizes the regression coefficients or weights. It measures how a linear regression model is performing.

→ We can use the cost function to find the accuracy of the **mapping function**, which maps the input variable to the output variable. This mapping function is also known as **Hypothesis function**.

→ For Linear Regression, we use the **Mean Squared Error (MSE)** cost function, which is the average of squared error occurred between the predicted values and actual values.

→ It can be written as:

For the above linear equation, MSE can be calculated as:

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - (a_1 x_i + a_0))^2$$

where,

N=Total number of observation

Yi = Actual value

(a1xi+a0)= Predicted value.

Residuals: The distance between the actual value and predicted values is called residual. If the observed points are far from the regression line, then the residual will be high, and so cost function will high. If the scatter points are close to the regression line, then the residual will be small and hence the cost function.

Gradient Descent:

→ Gradient descent is used to minimize the MSE by calculating the gradient of the cost function.

→ A regression model uses gradient descent to update the coefficients of the line by reducing the cost function.

→ It is done by a random selection of values of coefficient and then iteratively update the values to reach the minimum cost function.

Model Performance:

The Goodness of fit determines how the line of regression fits the set of observations. The process of finding the best model out of various models is called **optimization**. It can be achieved by below method:

1. **R-squared method:**

→ R−squared is a statistical method that determines the goodness of fit.

→ It measures the strength of the relationship between the dependent and independent variables on a scale of 0−100%.

→ The high value of R–square determines the less difference between the predicted values and actual values and hence represents a good model.

→ It is also called a **coefficient of determination,** or **coefficient of multiple determination** for multiple regression.

→ It can be calculated from the below formula:

$$\text{R-squared} = \frac{\text{Explained variation}}{\text{Total Variation}}$$

---

2. LEAST SQUARED METHOD

→ The least–squares regression method is a technique commonly used in Regression Analysis. It is a mathematical method used to find the best fit line that represents the relationship between an independent and dependent variable.

→ To understand the least–squares regression method lets get familiar with the concepts involved in formulating the line of best fit.

What is the Line Of Best Fit?

→ Line of best fit is drawn to represent the relationship between 2 or more variables. To be more specific, the best fit line is drawn across a scatter plot of data points in order to represent a relationship between those data points.

→ Regression analysis makes use of mathematical methods such as least squares to obtain a definite relationship between the predictor variable (s) and the target variable.

→ The least–squares method is one of the most effective ways used to draw the line of best fit. It is based on the idea that the square of the errors obtained must be minimized to the most possible extent and hence the name least squares method.

→ If we were to plot the best fit line that shows the depicts the sales of a company over a period of time, it would look something like this:

Notice that the line is as close as possible to all the scattered data points. This is what an ideal best fit line looks like. To better understand the whole process let's see how to calculate the line using the Least Squares Regression.

Steps to calculate the Line of Best Fit

To start constructing the line that best depicts the relationship between variables in the data, we first need to get our basics right. Take a look at the equation below:

$$y = mx + c$$

Surely, you've come across this equation before. It is a simple equation that represents a straight line along 2-Dimensional data, i.e. x−axis and y−axis. To better understand this, let's break down the equation:

→ y: dependent variable

→ m: the slope of the line

→ x: independent variable

→ c: y−intercept

So, the aim is to calculate the values of slope, y−intercept and substitute the corresponding 'x' values in the equation in order to derive the value of the dependent variable.

Let's see how this can be done.

As an assumption, let's consider that there are 'n' data points.

Step 1: Calculate the slope 'm' by using the following formula:

$$m = \frac{n\sum xy - (\Sigma x)(\Sigma y)}{n\Sigma x^2 - (\Sigma x)^2}$$

Step 2: Compute the y−intercept (the value of y at the point where the line crosses the yaxis):

Step 3: Substitute the values in the final equation:

$$y = mx + c$$

Now let's look at an example and see how you can use the least−squares regression method to compute the line of best fit.

Least Squares Regression Example

Consider an example. Tom who is the owner of a retail shop, found the price of different T−shirts vs the number of T−shirts sold at his shop over a period of one week. He tabulated this like shown below:

Price of T-shirts in dollars (x)	# of T-shirts sold (y)
2	4
3	5
5	7
7	10
9	15

Let us use the concept of least squares regression to find the line of best fit for the above data.

Step 1: Calculate the slope 'm' by using the following formula:

$$m = \frac{n\sum xy - (\Sigma x)(\Sigma y)}{n\Sigma x^2 - (\Sigma x)^2}$$

After you substitute the respective values, m = 1.518 approximately.

Step 2: Compute the y−intercept value

$$c = y - mx$$

After you substitute the respective values, c = 0.305 approximately.

Step 3: Substitute the values in the final equation

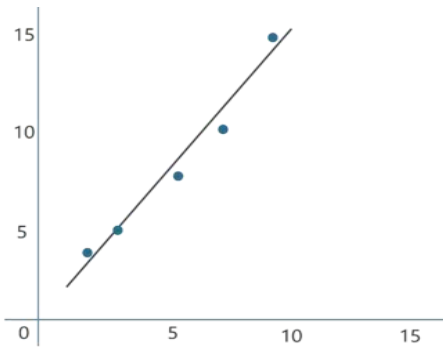
$$y = mx + c$$

Once you substitute the values, it should look something like this:

$$c = y - mx$$

Price of T-shirts in dollars (x)	# of T-shirts sold (y)	Y=mx+c	error
2	4	3.3	-0.67
3	5	4.9	-0.14
5	7	7.9	0.89
7	10	10.9	0.93
9	15	13.9	-1.03

Let's construct a graph that represents the **y=mx + c** line of best fit:



Now Tom can use the above equation to estimate how many T−shirts of price $8 can he sell at the retail shop.

y = 1.518 x 8 + 0.305 = 12.45 T−shirts

This comes down to 13 T−shirts! That's how simple it is to make predictions using Linear Regression.

3. LASSO REGRESSION

Whenever we hear the term "regression," two things that come to mind are **linear regression** and logistic regression. Even though the **logistic regression** falls under the classification algorithms category still it buzzes in our mind.

These two topics are quite famous and are the basic introduction topics in Machine Learning. There are other types of regression, like

→ Lasso regression,

→ Ridge regression,

→ Polynomial regression,

→ Stepwise regression,

→ ElasticNet regression

The above-mentioned techniques are majorly used in regression kind of analytical problems. When we **increase** the degree of freedom (increasing polynomials in the equation) for **regression models**, they tend to overfit. Using the **regularization techniques** we can overcome the overfitting issue.

Two popular methods for that is lasso and ridge regression. In our **ridge regression article** we explained the theory behind the ridge regression also we learned the implementation part in python.

What Is Regression?

Regression is a statistical technique used to determine the relationship between one dependent variable and one or many independent variables. In simple words, a regression analysis will tell you how your result varies for different factors.

For example,

What determines a person's salary?

Many factors, like educational qualification, experience, skills, job role, company, etc., play a role in salary.

You can use regression analysis to predict the dependent variable – salary using the mentioned factors.

$$y = mx+c$$

Do you remember this equation from our school days?

It is nothing but a linear regression equation. In the above equation, the dependent variable estimates the independent variable.

In mathematical terms,

→ Y is the dependent value,

→ X is the independent value,

→ m is the slope of the line,

→ c is the constant value.

The same equation terms are called slighted differently in **machine learning** or the statistical world.

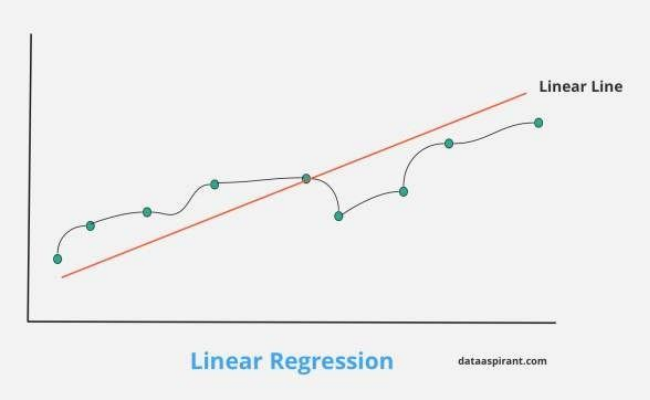
→ Y is the **predicted** value,

→ X is **feature** value,

→ m is **coefficients** or weights,

→ c is the **bias** value.

The line in the above graph represents the **linear regression model**. You can see how well the model fits the data. It looks like a good model, but sometimes the model fits the data too much, resulting in overfitting.



To create the line (red) using the actual value, the regression model will iterate and recalculate the **m**(coefficient) and **c** (bias) values while trying to **reduce the loss values** with the proper **loss function**.

The model will have low bias and high variance due to overfitting. The model fit is good in the training data, but it will not give good test data predictions. Regularization comes into play to tackle this issue.

What Is Regularization?

Regularization solves the **problem of overfitting**. Overfitting causes low model accuracy. It happens when the model learns the data as well as the noises in the training set.

Noises are random datum in the training set which don't represent the actual properties of the data.

$$Y \approx C0 + C1X1 + C2X2 + \ldots + CpXp$$

Y represents the **dependent** variable, X represents the **independent** variables and C represents the coefficient estimates for different variables in the above linear regression equation.

The model fitting involves a loss function known as the sum of squares. The coefficients in the equation are chosen in a way to **reduce** the loss function to a minimum value. Wrong coefficients get selected if there is a lot of irrelevant data in the training set.

Definition Of Lasso Regression

→ Lasso regression is like linear regression, but it uses a technique **"shrinkage"** where the coefficients of determination are shrunk towards **zero**. Linear regression gives you regression coefficients as observed in the dataset.

→ The lasso regression allows you to shrink or regularize these coefficients to avoid overfitting and make them work better on different datasets.

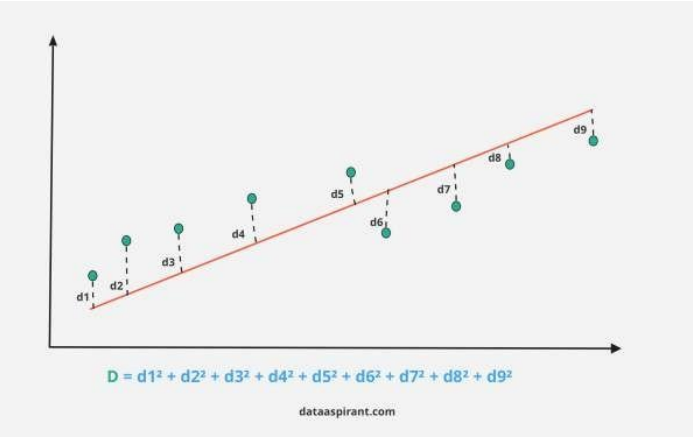
→ This type of regression is used when the dataset shows high multicollinearity or when you want to automate variable elimination and **feature selection**.

When To Use Lasso Regression?

Choosing a model depends on the dataset and the problem statement you are dealing with. It is essential to understand the dataset and how features interact with each other.

Lasso regression **penalizes** less important features of your dataset and makes their respective coefficients zero, thereby eliminating them. Thus it provides you with the benefit of feature selection and simple model creation. So, if the dataset has high dimensionality and high correlation, lasso regression can be used.

The Statistics of Lasso Regression



Statistics of lasso regression

→ d1, d2, d3, etc., represents the distance between the actual data points and the model line in the above graph. Least−squares is the sum of squares of the **distance between the points** from the plotted curve.

→ In linear regression, the best model is chosen in a way to **minimize** the least−squares. While performing lasso regression, we add a penalizing factor to the least−squares.

→ That is, the model is chosen in a way to reduce the below loss function to a minimal value.

D = least-squares + lambda * summation (absolute values of the magnitude of the coefficients)

→ Lasso regression penalty consists of all the estimated parameters. Lambda can be any value between zero to infinity. This value decides how aggressive regularization is performed. It is usually chosen using cross−validation.

→ Lasso penalizes the sum of absolute values of coefficients. As the lambda value increases, coefficients decrease and eventually **become zero**. This way, lasso regression eliminates insignificant variables from our model. Our regularized model may have a slightly high bias than linear regression but less variance for future predictions.

4. DECISION TREES

Tree based methods – Decision Trees

→ Tree−based machine learning methods are among the most commonly used supervised learning methods. They are constructed by two entities; branches and nodes.

→ Tree−based ML methods are built by recursively splitting a training sample, using different features from a dataset at each node that splits the data most effectively.

→ The splitting is based on learning simple decision rules inferred from the training data. Generally, tree−based ML methods are simple and intuitive; to predict a class label or value, we start from the top of the tree or the root and, using branches, go to the nodes by comparing features on the basis of which will provide the best split.

→ Tree−based methods also use the mean for continuous variables or mode for categorical variables when making predictions on training observations in the regions they belong to. Since the set of rules used to segment the predictor space can be summarized in a visual representation with branches that show all the possible outcomes, these approaches are commonly referred to as decision tree methods.

→ The methods are flexible and can be applied to either classification or regression problems. *Classification and Regression Trees* (CART) is a commonly used term by Leo Breiman, referring to the flexibility of the methods in solving both linear and non−linear predictive modeling problems.

Types of Decision Trees

Decision trees can be classified based on the type of target or response variable.

i. Classification Trees

The default type of decision trees, used when the response variable is categorical—i.e. predicting whether a team will win or lose a game.

ii. Regression Trees

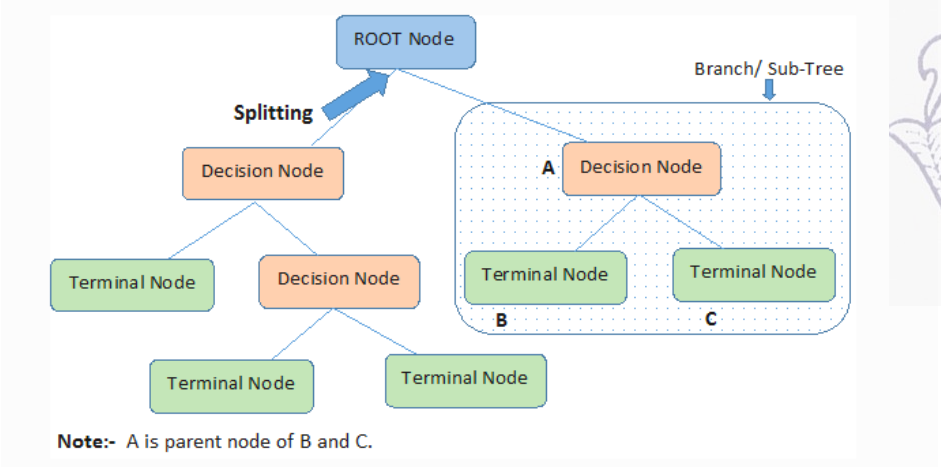
Used when the target variable is continuous or numerical in nature—i.e. predicting house prices based on year of construction, number of rooms, etc.

Advantages of Tree-based Machine Learning Methods

1. Interpretability: Decision tree methods are easy to understand even for non− technical people.
2. The data type isn't a constraint, as the methods can handle both categorical and numerical variables.
3. Data exploration — Decision trees help us easily identify the most significant variables and their correlation.

Disadvantages of Tree-based Machine Learning Methods

1. Large decision trees are complex, time−consuming and less accurate in predicting outcomes.
2. Decision trees don't fit well for continuous variables, as they lose important information when segmenting the data into different regions.



i) Root node — this represents the entire population or the sample, which gets divided into two or more homogenous subsets.

ii) Splitting — subdividing a node into two or more sub–nodes.

iii) ***Decision node*** — this is when a sub–node is divided into further sub–nodes.

iv) Leaf/Terminal node — this is the final/last node that we consider for our model output. It cannot be split further.

v) Pruning — removing unnecessary sub–nodes of a decision node to combat overfitting.

vi) Branch/Sub-tree — the sub–section of the entire tree.

vii) Parent and Child node — a node that's subdivided into a sub–node is a parent, while the sub–node is the child node.

CART (Classification and Regression Tree (CART)

CART (Classification And Regression Tree) is a variation of the decision tree algorithm. It can handle both classification and regression tasks. Scikit–Learn uses the Classification And Regression Tree (CART) algorithm to train Decision Trees (also called "growing" trees). CART was first produced by Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone in 1984.

CART Algorithm

→ CART is a predictive algorithm used in Machine learning and it explains how the target variable's values can be predicted based on other matters.

→ It is a decision tree where each fork is split into a predictor variable and each node has a prediction for the target variable at the end.

→ In the decision tree, nodes are split into sub–nodes on the basis of a threshold value of an attribute.

→ The root node is taken as the training set and is split into two by considering the best attribute and threshold value.

→ Further, the subsets are also split using the same logic. This continues till the last pure sub–set is found in the tree or the maximum number of leaves possible in that growing tree.

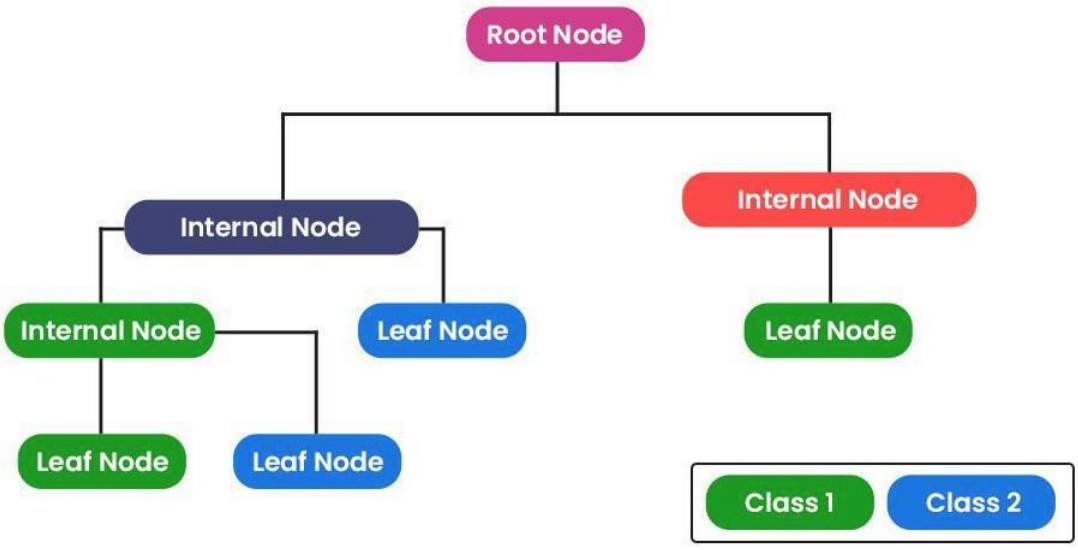
The CART algorithm works via the following process:

→ The best split point of each input is obtained.

→ Based on the best split points of each input in Step 1, the new "best" split point is identified.

→ Split the chosen input according to the "best" split point.

→ Continue splitting until a stopping rule is satisfied or no further desirable splitting is available.



CART algorithm uses Gini Impurity to split the dataset into a decision tree .It does that by searching for the best homogeneity for the sub nodes, with the help of the Gini index criterion.

Gini index/Gini impurity

→ The Gini index is a metric for the classification tasks in CART.

→ It stores the sum of squared probabilities of each class.

→ It computes the degree of probability of a specific variable that is wrongly being classified when chosen randomly and a variation of the Gini coefficient.

→ It works on categorical variables, provides outcomes either "successful" or "failure" and hence conducts binary splitting only.

The degree of the Gini index varies from 0 to 1,

→ Where 0 depicts that all the elements are allied to a certain class, or only one class exists there.

→ The Gini index of value 1 signifies that all the elements are randomly distributed across various classes &

→ A value of 0.5 denotes the elements are uniformly distributed into some classes.

Mathematically, we can write Gini Impurity as follows:

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

where pi is the probability of an object being classified to a particular class.

Classification tree

A classification tree is an algorithm where the target variable is categorical. The algorithm is then used to identify the "Class" within which the target variable is most likely to fall. Classification trees are used when the dataset needs to be split into classes that belong to the response variable (like yes or no).

Regression tree

A Regression tree is an algorithm where the target variable is continuous and the tree is used to predict its value. Regression trees are used when the response variable is continuous. For example, if the response variable is the temperature of the day.

CART model representation

CART models are formed by picking input variables and evaluating split points on those variables until an appropriate tree is produced.

Steps to create a Decision Tree using the CART algorithm:

→ **Greedy algorithm**: In this, The input space is divided using the Greedy method which is known as a recursive binary spitting. This is a numerical method within which all of the values are aligned and several other split points are tried and assessed using a cost function.

→ **Stopping Criterion:** As it works its way down the tree with the training data, the recursive binary splitting method described above must know when to stop splitting. The most frequent halting method is to utilize a minimum amount of training data allocated to every leaf node. If the count is smaller than the specified threshold, the split is rejected and also the node is considered the last leaf node.

→ **Tree pruning:** Decision tree's complexity is defined as the number of splits in the tree. Trees with fewer branches are recommended as they are simple to grasp and less prone to cluster the data. Working through each leaf node in the tree and evaluating the effect of deleting it using a hold−out test set is the quickest and simplest pruning approach.

→ **Data preparation for the CART:** No special data preparation is required for the CART algorithm.

Advantages of CART

→ Results are simplistic.

→ Classification and regression trees are Nonparametric and Nonlinear.

→ Classification and regression trees implicitly perform feature selection.

→ Outliers have no meaningful effect on CART.

→ It requires minimal supervision and produces easy−to−understand models.

Limitations of CART

→ Overfitting.

→ High Variance.

→ low bias.

→ the tree structure may be unstable.

Applications of the CART algorithm

→ For quick Data insights.

→ In Blood Donors Classification.

→ For environmental and ecological data.

→ In the financial sectors.

XIII. LOGISTIC REGRESSION

different in the ways that they learn about the solution; in essence they aim to do the same thing: find decision boundaries that can be used to separate out the different classes. Given the features that are used as inputs to the classifier, we need to identify some values of those features that will enable us to decide which class the current input is in. Figure 1.5 shows a set of 2D inputs with three different classes shown, and two different decision boundaries; on the left they are straight lines, and are therefore simple, but don't categorise as well as the non-linear curve on the right.

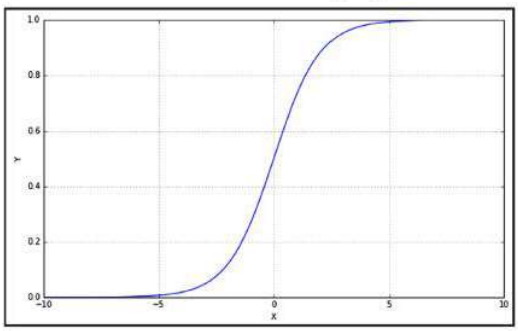
Now that we have seen these two types of problem, let's take a look at the whole process of machine learning from the practitioner's viewpoint.

7. Logistic Regression-

Even if called regression, this is a classification method which is based on the probability for a sample to belong to a class. As our probabilities must be continuous in R and bounded between (0, 1), it's necessary to introduce a threshold function to filter the term z. The name logistic comes from the decision to use the sigmoid (or logistic) function:

$$\sigma(z) = \frac{1}{1+e^{-z}} \text{ which becomes } \sigma(\bar{x};\bar{w}) = \frac{1}{1+e^{-\bar{x}\cdot\bar{w}}}$$

A partial plot of this function is shown in the following figure:



As you can see, the function intersects $x=0$ in the ordinate 0.5, and $y<0.5$ for $x<0$ and $y>0.5$ for $x>0$. Moreover, its domain is R and it has two asymptotes at 0 and 1. So, we can define the probability for a sample to belong to a class (from now on, we'll call them 0 and 1) as:

$$P(y|\bar{x}) = \sigma(\bar{x};\bar{w})$$

At this point, finding the optimal parameters is equivalent to maximizing the log-likelihood given the output class:

$$L(\bar{w};y) = \log P(y|\bar{w}) = \sum_i \log P(y_i|\bar{x}_i,\bar{w})$$

Therefore, the optimization problem can be expressed, using the indicator notation, as the minimization of the loss function:

$$J(\bar{w}) = -\sum_i \log P(y_i|\bar{x}_i,\bar{w}) = -\sum_i (y_i \log \sigma(z_i) + (1-y_i)\log(1-\sigma(z_i)))$$

If $y=0$, the first term becomes null and the second one becomes $\log(1-x)$, which is the log-probability of the class 0. On the other hand, if $y=1$, the second term is 0 and the first one represents the log-probability of x. In this way, both cases are embedded in a single expression. In terms of information theory, it means minimizing the cross-entropy between a target distribution and an approximated one:

$$H(X) = -\sum_{x \in X} p(x) log_2 q(x)$$

In particular, if log_2 is adopted, the functional expresses the number of extra bits requested to encode the original distribution with the predicted one. It's obvious that when $J(w) = 0$, the two distributions are equal. Therefore, minimizing the cross-entropy is an elegant way to optimize the prediction error when the target distributions are categorical.

We can generalize linear regression to the (binary) classification setting by making two changes. First we replace the Gaussian distribution for y with a **Bernoulli** distribution[9],which is more appropriate for the case when the response is binary, $y \in \{0, 1\}$. That is, we use

$$p(y|\mathbf{x}, \mathbf{w}) = \text{Ber}(y|\mu(\mathbf{x})) \tag{1.8}$$

where $\mu(\mathbf{x}) = \mathbb{E}[y|\mathbf{x}] = p(y = 1|\mathbf{x})$. Second, we compute a linear combination of the inputs, as before, but then we pass this through a function that ensures $0 \leq \mu(\mathbf{x}) \leq 1$ by defining

$$\mu(\mathbf{x}) = \text{sigm}(\mathbf{w}^T \mathbf{x}) \tag{1.9}$$

where $\text{sigm}(\eta)$ refers to the **sigmoid** function, also known as the **logistic** or **logit** function. This is defined as

$$\text{sigm}(\eta) \triangleq \frac{1}{1 + \exp(-\eta)} = \frac{e^\eta}{e^\eta + 1} \tag{1.10}$$

The term "sigmoid" means S-shaped: see Figure 1.19(a) for a plot. It is also known as a **squashing function**, since it maps the whole real line to $[0, 1]$, which is necessary for the output to be interpreted as a probability.

Putting these two steps together we get

$$p(y|\mathbf{x}, \mathbf{w}) = \text{Ber}(y|\text{sigm}(\mathbf{w}^T \mathbf{x})) \tag{1.11}$$

This is called **logistic regression** due to its similarity to linear regression (although it is a form of classification, not regression!).

A simple example of logistic regression is shown in Figure 1.19(b), where we plot

$$p(y_i = 1|x_i, \mathbf{w}) = \text{sigm}(w_0 + w_1 x_i) \tag{1.12}$$

where x_i is the SAT[10] score of student i and y_i is whether they passed or failed a class. The solid black dots show the training data, and the red circles plot $p(y = 1|\mathbf{x}_i, \hat{\mathbf{w}})$, where $\hat{\mathbf{w}}$ are the parameters estimated from the training data (we discuss how to compute these estimates in Section 8.3.4).

If we threshold the output probability at 0.5, we can induce a **decision rule** of the form

$$\hat{y}(x) = 1 \iff p(y = 1|\mathbf{x}) > 0.5 \tag{1.13}$$

By looking at Figure 1.19(b), we see that $\text{sigm}(w_0 + w_1 x) = 0.5$ for $x \approx 545 = x^*$. We can imagine drawing a vertical line at $x = x^*$; this is known as a decision boundary. Everything to the left of this line is classified as a 0, and everything to the right of the line is classified as a 1.

We notice that this decision rule has a non-zero error rate even on the training set. This is because the data is not **linearly separable**, i.e., there is no straight line we can draw to separate the 0s from the 1s. We can create models with non-linear decision boundaries using basis function expansion, just as we did with non-linear regression. We will see many examples of this later in the book.

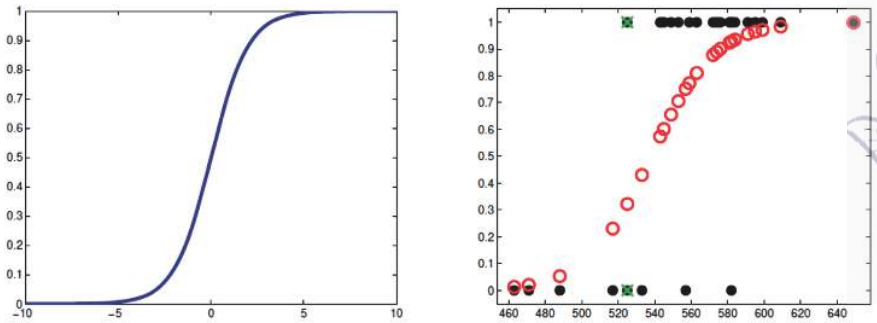


Figure 1.19 (a) The sigmoid or logistic function. We have $\text{sigm}(-\infty) = 0$, $\text{sigm}(0) = 0.5$, and $\text{sigm}(\infty) = 1$. Figure generated by `sigmoidPlot`. (b) Logistic regression for SAT scores. Solid black dots

8. Gradient Linear Models

In likelihood-based classification, the parameters were the sufficient statistics of $p(\boldsymbol{x}|C_i)$ and $P(C_i)$, and the method we used to estimate the parameters is maximum likelihood. In the discriminant-based approach, the parameters are those of the discriminants, and they are optimized to minimize the classification error on the training set. When $\boldsymbol{w}$ denotes the set of parameters and $E(\boldsymbol{w}|\mathcal{X})$ is the error with parameters $\boldsymbol{w}$ on the given training set $\mathcal{X}$, we look for

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w}|\mathcal{X})$$

GRADIENT DESCENT
GRADIENT VECTOR

In many cases, some of which we will see shortly, there is no analytical solution and we need to resort to iterative optimization methods, the most commonly employed being that of *gradient descent*. When $E(\boldsymbol{w})$ is a differentiable function of a vector of variables, we have the *gradient vector* composed of the partial derivatives

$$\nabla_w E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_d}\right]^T$$

and the *gradient descent* procedure to minimize E starts from a random $\boldsymbol{w}$, and at each step, updates $\boldsymbol{w}$, in the opposite direction of the gradient

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}, \forall i \quad (10.16)$$

$$w_i = w_i + \Delta w_i \quad (10.17)$$

where η is called the *stepsize*, or *learning factor*, and determines how much to move in that direction. Gradient ascent is used to maximize a function and goes in the direction of the gradient. When we get to a minimum (or maximum), the derivative is 0 and the procedure terminates. This indicates that the procedure finds the nearest minimum that can be a local minimum, and there is no guarantee of finding the global minimum unless the function has only one minimum. The use of a good value for η is also critical; if it is too small, the convergence may be too slow, and a large value may cause oscillations and even divergence.

UNIT IV SUPERVISED LEARNING

Neural Network: Introduction, Perceptron Networks – Adaline – Back propagation networks –

Decision Tree: Entropy – Information gain – Gini impurity – classification algorithm – Rule based Classification – **Naïve Bayesian classification – Support Vector Machines (SVM)**

1. NEURAL NETWORK

1.1 INTRODUCTION

Neural Networks is one of the most significant discoveries in history. Neural Networks can solve problems that can't be solved by algorithms:

→ Medical Diagnosis

→ Face Detection

→ Voice Recognition

Neural Networks is the essence of **Deep Learning**.

The Deep Learning Revolution

The deep learning revolution started around 2010. Since then, Deep Learning has solved many "unsolvable" problems. The deep learning revolution was not started by a single discovery. It more or less happened when several needed factors were ready:

→ Computers were fast enough

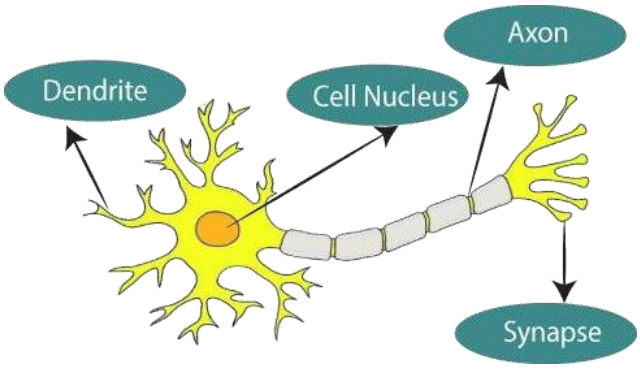
→ Computer storage was big enough

→ Better training methods were invented

→ Better tuning methods were invented

Neurons

Scientists agree that our brain has around 100 billion neurons. These neurons have hundreds of billions of connections between them.



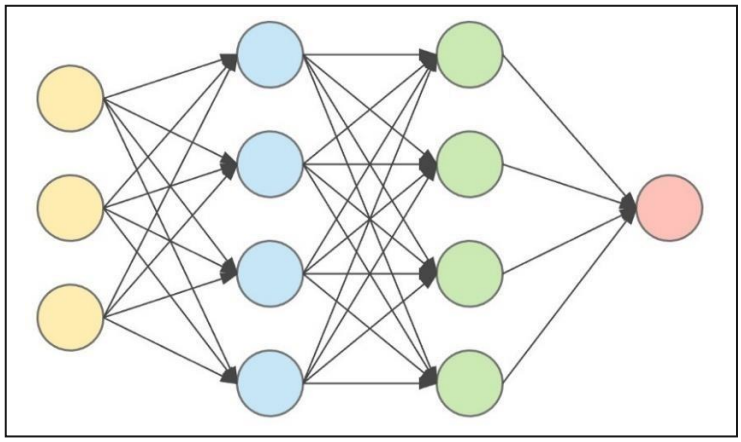
Neurons (aka Nerve Cells) are the fundamental units of our brain and nervous system. The neurons are responsible for receiving input from the external world, for sending output (commands to our muscles), and for transforming the electrical signals in between.

Neural Networks

Artificial Neural Networks are normally called Neural Networks (NN). Neural networks are in fact multi−layer **Perceptron's**. The perceptron defines the first step into multi− layered neural networks.

The Neural Network Model

Input data (Yellow) are processed against a hidden layer (Blue) and modified against another hidden layer (Green) to produce the final output (Red).



"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

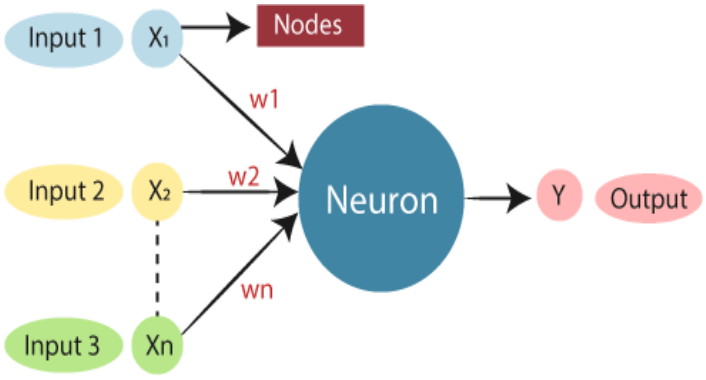
→ E: Experience (the number of times).

→ T: The Task (driving a car).

→ P: The Performance (good or bad).

The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.

The typical Artificial Neural Network looks something like the given figure.



Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

Relationship between Biological neural network and artificial neural network:

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

1.2 PERCEPTRON NETWORKS

→ Perceptron is Machine Learning algorithm for supervised learning of various binary classification tasks. Further, ***Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence***.

→ Perceptron model is also treated as one of the best and simplest types of Artificial Neural networks. However, it is a supervised learning algorithm of binary classifiers. Hence, we can consider it as a single−layer neural network with four main parameters, i.e., **input values, weights and Bias, net sum, and an activation function.**

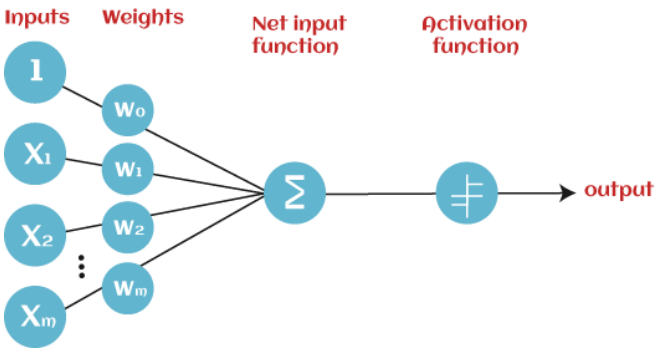
What is Binary classifier in Machine Learning?

→ In Machine Learning, binary classifiers are defined as the function that helps in deciding whether input data can be represented as vectors of numbers and belongs to some specific class.

→ Binary classifiers can be considered as linear classifiers. In simple words, we can understand it as a ***classification algorithm that can predict linear predictor function in terms of weight and feature vectors.***

Basic Components of Perceptron

Mr. Frank Rosenblatt invented the perceptron model as a binary classifier which contains three main components. These are as follows:



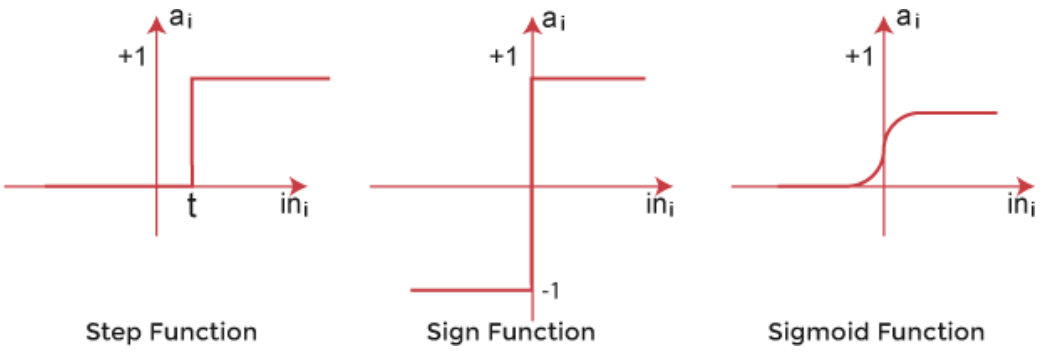
- **Input Nodes or Input Layer: -** This is the primary component of Perceptron which accepts the initial data into the system for further processing. Each input node contains a real numerical value.
- **Wight and Bias: -** Weight parameter represents the strength of the connection between units. This is another most important parameter of Perceptron components. Weight is directly proportional to the strength of the associated input neuron in deciding the output. Further, Bias can be considered as the line of intercept in a linear equation.
- **Activation Function: -** These are the final and 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 functions:

→ Sign function

→ Step function, and

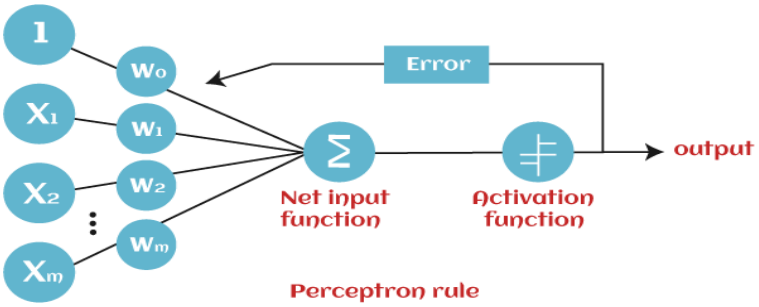
→ Sigmoid function



The data scientist uses the activation function to take a subjective decision based on various problem statements and forms the desired outputs. Activation function may differ (e.g., Sign, Step, and Sigmoid) in perceptron models by checking whether the learning process is slow or has vanishing or exploding gradients.

How does Perceptron work?

In Machine Learning, Perceptron is considered as a single−layer neural network that consists of four main parameters named input values (Input nodes), weights and Bias, net sum, and an activation function. The perceptron model begins with the multiplication of all input values and their weights, then adds these values together to create the weighted sum. Then this weighted sum is applied to the activation function 'f' to obtain the desired output.



This activation function is also known as the **step function** and is represented by **'f'**.

This step function or Activation function plays a vital role in ensuring that output is mapped between required values (0,1) or (−1,1). It is important to note that the weight of input is indicative of the strength of a node. Similarly, an input's bias value gives the ability to shift the activation function curve up or down.

Perceptron model works in two important steps as follows:

Step-1 - In the first step first, multiply all input values with corresponding weight values and then add them to determine the weighted sum. Mathematically, we can calculate the weighted sum as follows:

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

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

$$\sum wi*xi + b$$

Step-2 - In the second step, an activation function is applied with the above−mentioned weighted sum, which gives us output either in binary form or a continuous value as follows: $Y = f(\sum wi*xi + b)$

Types of Perceptron Models

Based on the layers, Perceptron models are divided into two types. These are as follows:

1. Single−layer Perceptron Model
2. Multi−layer Perceptron model

Single Layer Perceptron Model:

→ This is one of the easiest Artificial neural networks (ANN) types. A single−layered perceptron model consists feed−forward network and also includes a threshold transfer function inside the model. The main objective of the single−layer perceptron model is to analyze the linearly separable objects with binary outcomes.

→ In a single layer perceptron model, its algorithms do not contain recorded data, so it begins with inconstantly allocated input for weight parameters. Further, it sums up all inputs (weight). After adding all inputs, if the total sum of all inputs is more than a pre−determined value, the model gets activated and shows the output value as +1.

→ If the outcome is same as pre−determined or threshold value, then the performance of this model is stated as satisfied, and weight demand does not change. However, this model consists of a few discrepancies triggered when multiple weight inputs values are fed into the model. Hence, to find desired output and minimize errors, some changes should be necessary for the weights input.

"Single−layer perceptron can learn only linearly separable patterns."

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 layers.

→ The multi−layer perceptron model is also known as the Backpropagation algorithm, which executes in two stages as follows:

- **Forward Stage:** Activation functions start from the input layer 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 model's requirement. In this stage, the error between actual output and demanded originated backward on the output layer and ended on the input layer.

Hence, a multi−layered perceptron model has considered as multiple artificial neural networks having various layers in which activation function does not remain linear, similar to a single layer perceptron model. Instead of linear, activation function can be executed as sigmoid, TanH, ReLU, etc., for deployment.

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

Perceptron Function

Perceptron function "f(x)" can be achieved as output by multiplying the input 'x' with the learned weight coefficient 'w'.

Mathematically, we can express it as follows:

f(x)=1; if w.x+b>0 otherwise, f(x)=0

→ 'w' represents real−valued weights vector

→ 'b' represents the bias

→ 'x' represents a vector of input x values.

1.3 ADALINE

→ ADALINE, or Adaptive Linear Neuron, is a foundational artificial neural network model developed by Bernard Widrow and Ted Hoff in 1960. It is a single-layer network that learns to perform binary classification and linear regression.

→ An artificial neural network inspired by the human neural system is a network used to process the data which consist of three types of layer i.e input layer, the hidden layer, and the output layer.

→ The basic neural network contains only two layers which are the input and output layers.

→ The layers are connected with the weighted path which is used to find net input data.

→ In this section, we will discuss two basic types of neural networks Adaline which doesn't have any hidden layer, and Madaline which has one hidden layer.

Adaline (Adaptive Linear Neural):

→ A network with a single linear unit is called Adaline (Adaptive Linear Neural). A unit with a linear activation function is called a linear unit.

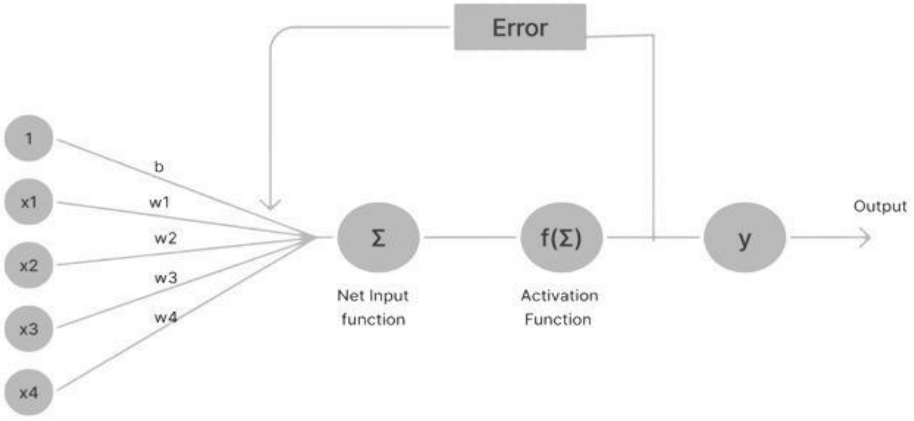
→ In Adaline, there is only one output unit and output values are bipolar (+1,-1). Weights between the input unit and output unit are adjustable.

→ It uses the delta rule i.e $wi(new)=wi(old)+\alpha(t-yin)xi$, where wi, yin and t are the weight, predicted output, and true value respectively.

→ The learning rule is found to minimize the mean square error between activation and target values.

→ Adaline consists of trainable weights, it compares actual output with calculated output, and based on error training algorithm is applied.

Workflow:



Adaline

First, calculate the net input to your Adaline network then apply the activation function to its output then compare it with the original output if both the equal, then give the output else send an error back to the network and update the weight according to the error which is calculated by the delta learning rule.

i.e $wi(new)=wi(old)+\alpha(t-yin)xi$, where wi, yin and t are the weight, predicted output, and true value respectively

Key Characteristics of ADALINE:

- **Linear Activation Function:**

  Unlike the Perceptron which uses a step function, ADALINE employs a linear activation function. This allows for the minimization of a continuous cost function, typically the Mean Squared Error (MSE), between the network's output and the desired target.

- **Supervised Learning with Delta Rule:**

  ADALINE is trained using supervised learning, meaning it learns from labeled data (input-output pairs). The weight adjustments are governed by the Delta Rule, also known as the Widrow-Hoff rule or Least Mean Square (LMS) rule. This rule utilizes gradient descent to iteratively update the weights and bias by minimizing the MSE.

- **Architecture:**

  - **Input Layer:** Receives the input features, often including a bias unit (a constant input, typically 1).

- **Weighted Summation:** Each input feature is multiplied by its corresponding weight, and these weighted inputs are summed along with the bias.
- **Linear Activation:** The summed value is passed through a linear activation function, which simply outputs the input value itself.
- **Output Layer:** Produces the final output, which is then compared to the target value.

- **Weight Update Mechanism:**

The core of ADALINE's learning lies in its weight and bias update rule. For each input example, the error (difference between target and actual output) is calculated. The weights and bias are then adjusted in proportion to this error, the learning rate (a hyperparameter controlling the step size of updates), and the input values, aiming to reduce the error in subsequent iterations.

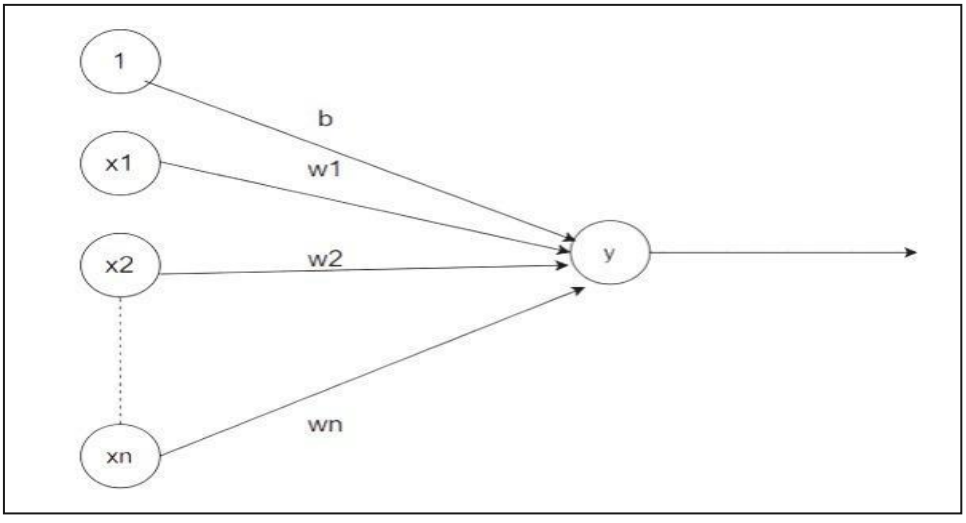
Significance in AI and Machine Learning:

→ ADALINE represents a crucial step in the evolution of neural networks.

→ Its use of a continuous activation function and gradient descent for weight updates laid the groundwork for more complex and powerful models, including multilayer perceptron's and deep neural networks.

→ While limited to linearly separable problems, its principles of error minimization and adaptive weight adjustment are fundamental to modern machine learning algorithms.

Architecture:



Adaline

In Adaline, all the input neuron is directly connected to the output neuron with the weighted connected path. There is a bias b of activation function 1 is present.

Algorithm:

Step 1: Initialize weight not zero but small random values are used. Set learning rate α.

Step 2: While the stopping condition is False do steps 3 to 7.

Step 3: for each training set perform steps 4 to 6.

Step 4: Set activation of input unit $x_i = s_i$ for (i=1 to n).

Step 5: compute net input to output unit

$yin=\sum wixi+b$ $yin=\sum wixi+b$

Here, b is the bias and n is the total number of neurons.

Step 6: Update the weights and bias for i=1 to n

$wi(new)=wi(old)+\alpha(t-yin)xi$

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

and calculate $error:(t-yin)2$

when the predicted output and the true value are the same then the weight will not change.

Step 7: Test the stopping condition. The stopping condition may be when the weight changes at a low rate or no change.

Implementations

Problem: Design OR gate using Adaline Network?

Solution:

- Initially**,** all weights are assumed to be small random values, say 0.1, and set learning rule to 0.1.
- Also, set the least squared error to 2.
- The weights will be updated until the total error is greater than the least squared error.

x_1	x_2	t
1	1	1
1	-1	1
-1	1	1
-1	-1	-1

- Calculate the net input $yin=\sum wixi+b$

  $yin=0.1\times1+0.1\times1+0.1=0.3$ (when $x_1=x_2=1$)
- Now compute, $(t-y_{in})=(1-0.3)=0.7$
- Now, update the weights and bias

  $wi(new)=wi(old)+\alpha(t-yin)xi$

  $w1(new)=0.1+0.1(1-0.3)1=0.17$

  $w2(new)=0.1+0.1(1-0.3)1=0.17$

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

  $b(new)=0.1+0.1(1-0.3)=0.17$
- calculate the error: $error=(t-yin)2=0.72=0.49$

Similarly, repeat the same steps for other input vectors and you will get.

x_1	x_2	t	y_{in}	$(t-y_{in})$	Δw_1	Δw_2	Δb	w_1 (0.1)	w_2 (0.1)	b (0.1)	$(t-y_{in})^2$
1	1	1	0.3	0.7	0.07	0.07	0.07	0.17	0.27	0.17	0.49
1	-1	1	0.17	0.83	0.083	-0.083	0.083	0.253	0.087	0.253	0.69
-1	1	1	0.087	0.913	-0.0913	0.0913	0.1004	0.1617	0.1783	0.3443	0.83
-1	-1	-1	0.0043	-1.0043	0.1004	0.1004	-0.1004	0.2621	0.2787	0.2439	1.01

This is epoch 1 where the total error is 0.49 + 0.69 + 0.83 + 1.01 = 3.02 so more epochs will run until the total error becomes less than equal to the least squared error i.e 2.

```
# Import necessary libraries
import numpy as np

#Adaline neural network
def Adaline(Input, Target, lr=0.2, stop=0.001):
    weight = np.random.random(Input.shape[1])
    bias   = np.random.random(1)

    Error=[stop +1]
    # check the stop condition for the network
    while Error[-1] > stop or Error[-1]-Error[-2] > 0.0001:
        error = []
        for i in range(Input.shape[0]):
            Y_input = sum(weight*Input[i]) + bias

            # Update the weight
            for j in range(Input.shape[1]):
                weight[j]=weight[j] + lr*(Target[i]-Y_input)*Input[i][j]

            # Update the bias
            bias=bias + lr*(Target[i]-Y_input)

            # Store squared error value
            error.append((Target[i]-Y_input)**2)
        # Store sum of square errors
        Error.append(sum(error))
        print('Error :',Error[-1])
    return weight, bias

# Input dataset
x = np.array([[1.0, 1.0, 1.0],
              [1.0, -1.0, 1.0],
              [-1.0, 1.0, 1.0],
              [-1.0, -1.0, -1.0]])
# Target values
t = np.array([1, 1, 1, -1])

w,b = Adaline(x, t, lr=0.2, stop=0.001)
print('weight :',w)
print('Bias :',b)
```

Output:

```
Error: [2.33228319]
Error: [1.09355784]
Error: [0.73680883]
Error: [0.50913731]
Error: [0.35233593]
Error: [0.24384625]
Error: [0.16876305]
Error: [0.11679891]
Error: [0.08083514]
Error: [0.05594504]
Error: [0.0387189]
Error: [0.02679689]
Error: [0.01854581]
Error: [0.01283534]
Error: [0.00888318]
Error: [0.00614795]
Error: [0.00425492]
Error: [0.00294478]
Error: [0.00203805]
Error: [0.00141051]
Error : [0.0009762]
weight: [0.01081771 0.01081771 0.98675106]
Bias: [0.01081771]
```

Predictions:

Predict from the evaluated weight and bias of Adaline

```
# Predict from the evaluated weight and bias of adaline
def prediction(X,w,b):
    y=[]
    for i in range(X.shape[0]):
        x = X[i]
        y.append(sum(w*x)+b)
    return y
prediction(x,w,b)
```

output:

```
[array([1.0192042]),
 array([0.99756877]),
 array([0.99756877]),
 array([-0.99756877])]
```

2. Madaline (Multiple Adaptive Linear Neuron):

→ The Madaline (supervised Learning) model consists of many Adaline in parallel with a single output unit.

→ The Adaline layer is present between the input layer and the Madaline layer hence Adaline layer is a hidden layer.

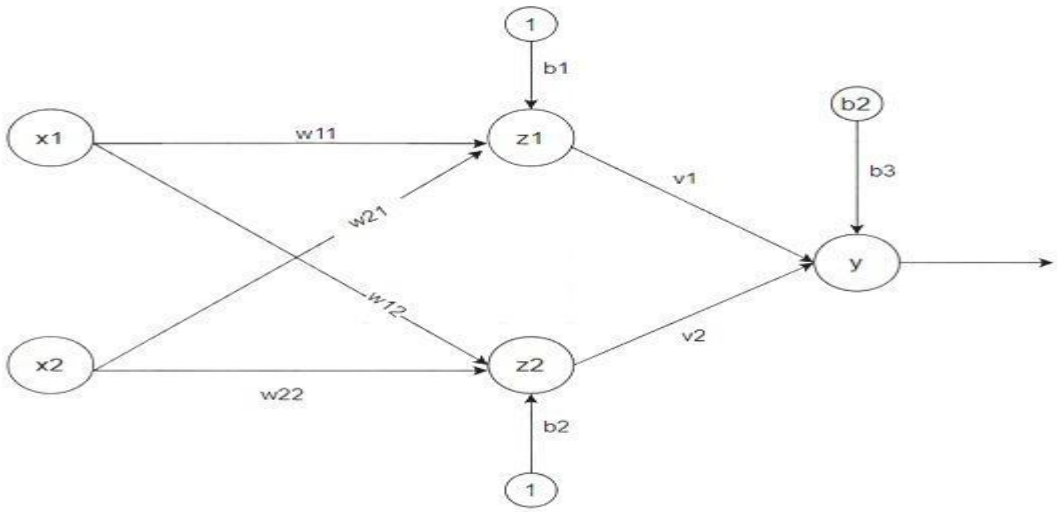
→ The weights between the input layer and the hidden layer are adjusted, and the weight between the hidden layer and the output layer is fixed.

→ It may use the majority vote rule, the output would have an answer either true or false.

→ Adaline and Madaline layer neurons have a bias of '1' connected to them.

→ use of multiple Adaline helps counter the problem of non-linear separability.

Architecture:



Madaline

There are three types of a layer present in Madaline First input layer contains all the input neurons, the Second hidden layer consists of an adaline layer, and weights between the input and hidden layers are adjustable and the third layer is the output layer the weights between hidden and output layer is fixed they are not adjustable.

Algorithm:

Step 1: Initialize weight and set learning rate α.

$v_1=v_2=0.5$, b=0.5, other weight may be a small random value.

Step 2: While the stopping condition is False do steps 3 to 9.

Step 3: for each training set perform steps 4 to 8.

Step 4: Set activation of input unit xi = si for (i=1 to n).

Step 5: compute net input of Adaline unit

$z_{in1} = b_1 + x_1w_{11} + x_2w_{21}$

$z_{in2} = b_2 + x_1w12 + x_2w_{22}$

Step 6: for output of remote Adaline unit using activation function given below:

Activation function f(z) =1 if $z \geq 0$ (and) (−1) if $z < 0$ 1 if $z \geq 0$ (*and*) (−1) if $z < 0$.

$z_1 = f(z_{in1})$

$z_2 = f(z_{in2})$

Step 7: Calculate the net input to output.

$y_{in} = b_3 + z_1v_1 + z_2v_2$

Apply activation to get the output of the net. $y = f(y_{in})$

Step 8: Find the error and do weight updation

if $t \neq y$ then t=1 update weight on z(j) unit whose next input is close to 0.

if t = y no updation

$w_{ij}(new) = w_{ij}(old) + \alpha(t - z_{inj})x_i$

$b_j(new) = b_j(old) + \alpha(t - z_{inj})$

if t=-1 then update weights on all unit z_k which have positive net input

Step 9: Test the stopping condition; weights change all number of epochs.

Problem: Using the Madaline network, implement XOR function with bipolar inputs and targets. Assume the required parameter for the training of the network.

Solution:

- Training pattern for XOR function:

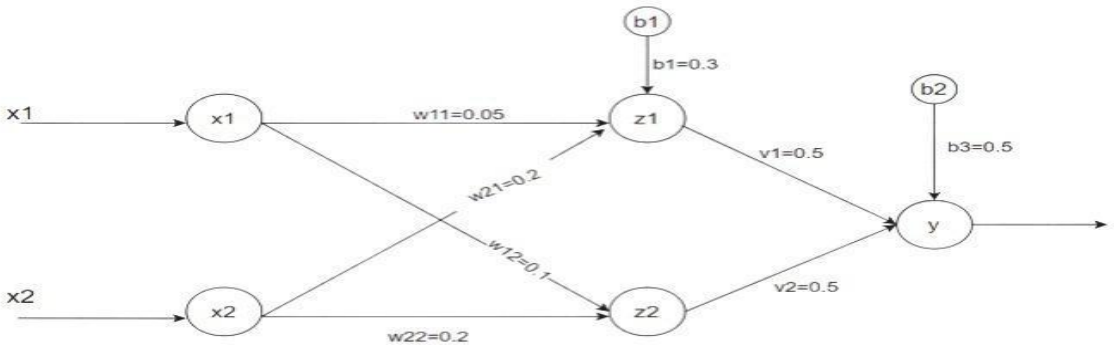
x_1	x_2	t
1	1	-1
1	-1	1
-1	1	1

- Initially, weights and bias are: Set $\alpha = 0.5$

$[w_{11} \quad w_{21} \quad b_1] = [0.05 \quad 0.2 \quad 0.3]$

$[w_{12} \quad w_{22} \quad b_2] = [0.1 \quad 0.2 \quad 0.15]$

$[v_1 \quad v_2 \quad v_3] = [0.5 \quad 0.5 \quad 0.5]$



Madaline

- for the first i/p & o/p pair from training data:

  $x_1 = 1$ $x_2 = 1$ $t = -1$ $\alpha = 0.5$

- Net input to the hidden unit:

$z_{in1} = b1 + x_1w_{11} + x_2w_{21} = 0.05 * 1 + 0.2 *1 + 0.3 = 0.55$

$z_{in2} = b2 + x_1w_{12} + x_2w_{22} = 0.1 * 1 + 0.2 *1 + 0.15 = 0.45$

- Apply the activation function f(z) to the net input

  $z_1 = f(z_{in1}) = f(0.55) = 1$

  $z_2 = f(z_{in2}) = f(0.45) = 1$

- computation for the output layer

$y_{in} = b3 + z_1v_1 + z_2v_2 = 0.5 + 1 *0.5 + 1*0.5 = 1.5$

$y=f(y_{in}) = f(1.5) = 1$

- Since (y=1) is not equal to (t=-1) update the weights and bias

$w_{ij}(new) =w_{ij}(old) + \alpha(t-z_{inj})x_i$

$b_j(new) = b_j(old) + \alpha(t-z_{inj})$

- $w_{11}(new) = w_{11}(old) + \alpha(t-z_{in1})x_1 = 0.05 + 0.5(-1-0.55) * 1 = -0.725$

  $w_{12}(new) = w_{12}(old) + \alpha(t-z_{in2})x_1 = 0.1 + 0.5(-1-0.45) * 1 = -0.625$

  $b1(new) = b1(old) + \alpha(t-z_{in1}) = 0.3 + 0.5(-1-0.55) = -0.475$

  $w_{21}(new) = w_{21}(old) + \alpha(t-z_{in1})x_2 = 0.2 + 0.5(-1-0.55) * 1 = -0.575$

  $w_{22}(new) = w_{22}(old) + \alpha(t-z_{in2})x_2 = 0.2 + 0.5(-1-0.45) * 1 = -0.525$

  $b_2(new) = b_2(old) + \alpha(t-z_{in2}) = 0.15 + 0.5(-1-0.45) = -0.575$

So, after epoch 1 weight like :

$[w_{11} \quad w_{21} \quad b_1] = [-0.725 \quad -0.57 \quad -0.475]$

$[w_{12} \quad w_{22} \quad b_2] = [-0.625 \quad -0.525 \quad -0.575]$

```
#Adaline neural network
import numpy as np
import pandas as pd
def activation_fn(z):
    if z>=0:
        return 1
    else:
        return -1
def Madaline(Input, Target, lr, epoch):
    weight = np.random.random((Input.shape[1],Input.shape[1]))
```


```
    bias   = np.random.random(Input.shape[1])
    w = np.array([0.5 for i in range(weight.shape[1])])
    b = 0.5
    k = 0
    while k<epoch:
        error = []
        z_input = np.zeros(bias.shape[0])
        z = np.zeros(bias.shape[0])
        for i in range(Input.shape[0]):
            for j in range(Input.shape[1]):
                z_input[j] = sum(weight[j]*Input[i]) + bias[j]
                z[j]= activation_fn(z_input[j])
            y_input = sum(z*w) +b
            y = activation_fn(y_input)
            # Update the weight & bias
            if y != Target[i]:
                for j in range(weight.shape[1]):
                    weight[j]= weight[j] + lr*(Target[i]-z_input[j])*Input[i][j]
                    bias[j]  = bias[j] + lr*(Target[i]-z_input[j])
            # Store squared error value
            error.append((Target[i]-z_input)**2)
        # compute sum of square error
        Error = sum(error)
        print(k,'>> Error :',Error)
        k+=1
    return weight, bias
# Input dataset
x = np.array([[1.0, 1.0, 1.0], [1.0, -1.0, 1.0],
              [-1.0, 1.0, 1.0], [-1.0, -1.0, -1.0]])
# Target values
t = np.array([1, 1, 1, -1])
w,b = Madaline(x, t, 0.0001, 3)
print('weight :',w)
print('Bias :',b)
```

Output:

```
0 >> Error: [4.51696958 1.53996419 2.66999799]
1 >> Error: [4.51696958 1.53996419 2.66999799]
2 >> Error: [4.51696958 1.53996419 2.66999799]
weight: [[0.1379015  0.86899587 0.7513866 ]
 [0.82302152 0.19126824 0.35891423]
 [0.52160397 0.1238258  0.88265076]]
Bias : [0.87199879 0.43476458 0.72613887]
```

Predictions:

Predict from the evaluated weight and bias of Madaline

```
def prediction(X, w,b):
   y =[]
   for i in range(X.shape[0]):
      x = X[i]
      z1 = x*w
      z_1 =[]
      for j in range(z1.shape[1]):
         z_1.append(activation_fn(sum(z1[j])+b[j]))
      y_in = sum(np.array(z_1)*np.array([0.5 for j in range(w.shape[1])])) + 0.5
      y.append(activation_fn(y_in))
   return y
prediction(x, w,b)
```

Output:

```
[1, 1, 1, -1]
```

The output of the Madaline is 100% correct.

1.4 BACK PROPAGATION NETWORKS

→ **Backpropagation** is the essence of neural network training.

→ It is the method of fine− tuning the weights of a neural network based on the error rate obtained in the previous epoch (i.e., iteration).

→ Proper tuning of the weights allows you to reduce error rates and make the model reliable by increasing its generalization.

→ Backpropagation in neural network is a short form for “backward propagation of error”.

→ It is a standard method of training artificial neural networks.

→ This method helps calculate the gradient of a loss function with respect to all the weights in the network.

How Backpropagation Algorithm Works

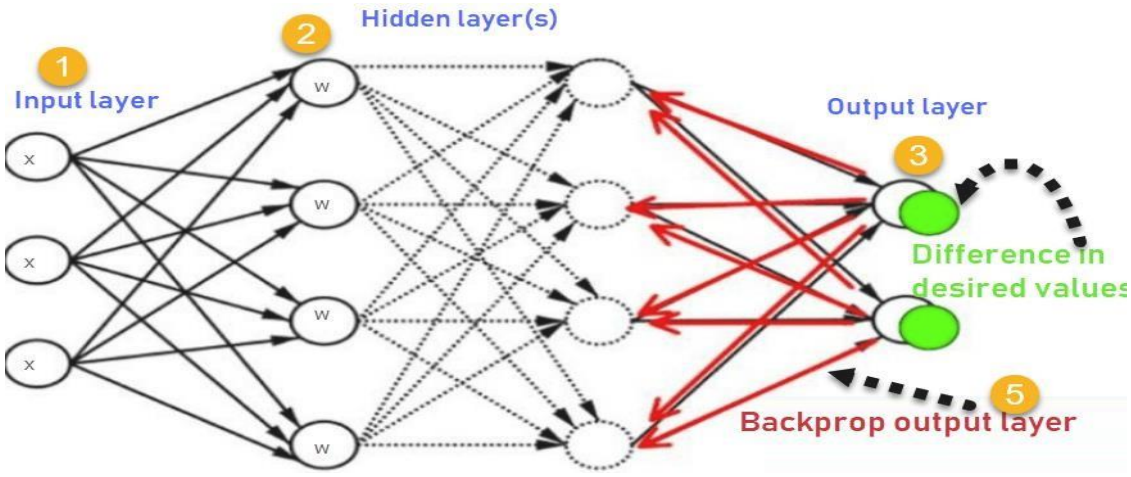
The Back propagation algorithm in neural network computes the gradient of the loss function for a single weight by the chain rule. It efficiently computes one layer at a time, unlike a native direct computation. It computes the gradient, but it does not define how the gradient is used. It generalizes the computation in the delta rule.

Consider the following Back propagation neural network example diagram to understand:

1. Inputs X, arrive through the preconnected path
2. Input is modeled using real weights W. The weights are usually randomly selected.
3. Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
4. Calculate the error in the outputs

   ErrorB= Actual Output – Desired Output
5. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

Keep repeating the process until the desired output is achieved



Why we need Backpropagation?

Most prominent advantages of Backpropagation are:

→ Backpropagation is fast, simple and easy to program

→ It has no parameters to tune apart from the numbers of input

→ It is a flexible method as it does not require prior knowledge about the network

→ It is a standard method that generally works well

→ It does not need any special mention of the features of the function to be learned.

Types of Backpropagation Networks

Two Types of Backpropagation Networks are:

- Static Back−propagation
- Recurrent Backpropagation

Static back-propagation:

It is one kind of backpropagation network which produces a mapping of a static input for static output. It is useful to solve static classification issues like optical character recognition.

Recurrent Backpropagation:

Recurrent Back propagation in data mining is fed forward until a fixed value is achieved. After that, the error is computed and propagated backward. The main difference between both of these methods is: that the mapping is rapid in static back−propagation while it is non-static in recurrent backpropagation.

---

2. DECISION TREE:

→ A decision tree is a supervised machine learning algorithm used for both classification and regression tasks.

→ It operates by recursively splitting a dataset into subsets based on the values of different features, creating a tree-like structure where each internal node represents a test on an attribute, each branch represents an outcome of the test, and each leaf node represents a class label (in classification) or a predicted value (in regression).

2.1 ENTROPY

→ Entropy is a measure of impurity or disorder within a set of data. In the context of decision trees, it quantifies the uncertainty in a node's class distribution.

→ A node with high entropy has a mixed distribution of classes, while a node with low entropy (approaching zero) is more "pure," meaning most instances belong to a single class.

→ The formula for entropy for a binary classification problem is:

Code

$$\text{Entropy} = -p(+)\log_2(p(+)) - p(-)\log_2(p(-))$$

where p(+) and p(-) are the probabilities of the positive and negative classes, respectively.

Entropy measures uncertainty in a node’s class distribution and originates from information theory. Higher entropy indicates greater disorder among class labels.

Formula:

$$\text{Entropy} = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Where *pi* represents the proportion of class *i* in the node.

Properties:

- Zero entropy corresponds to perfectly pure splits.
- Sensitive to small fluctuations across class ratios.
- Often yields balanced splits with meaningful boundaries.

2.2 INFORMATION GAIN

→ Information gain measures the reduction in entropy achieved by splitting a dataset based on a particular feature. It quantifies how much a feature contributes to clarifying the class labels.

→ Decision tree algorithms aim to maximize information gain at each split, selecting the feature that results in the purest child nodes (lowest entropy).

→ The formula for information gain is:

Code

Information Gain (S, A) = Entropy(S) - Σ((|Sv| / |S|) * Entropy(Sv))

where S is the parent node, A is the attribute being considered for splitting, Sv is the subset of S for which attribute A has value v, and |S| and |Sv| represent the number of elements in S and Sv, respectively.

Iterative Dichotomiser 3 (ID3)

→ **ID3 algorithm**, stands for Iterative Dichotomiser 3**,** is a classification **algorithm** that follows a greedy approach **of** building a decision tree by selecting a best attribute that yields maximum Information Gain (IG) or minimum Entropy (H)**.**

→ **Entropy** is a measure of the amount of uncertainty in the dataset S. Mathematical Representation of Entropy is shown here −

$$\mathbf{H(S) = \sum c \in C - p(c) \log 2\, p(c)}$$

Where,

- S − The current dataset for which entropy is being calculated(changes every iteration of the ID3 algorithm).
- C − Set of classes in S {example − C ={yes, no}}
- p(c) − The proportion of the number of elements in class c to the number of elements in set S.

In ID3, entropy is calculated for each remaining attribute. The attribute with the smallest entropy is used to split the set S on that particular iteration.

Entropy = 0 implies it is of pure class, that means all are of same category. Information Gain IG(A) tells us how much uncertainty in S was reduced after splitting set S on attribute A. Mathematical representation of Information gain is shown here −

$$I \qquad G(A, S) = H(S) - \sum t \in T\, p(t)\, H(t)$$

Where,

- H(S) − Entropy of set S.
- T − The subsets created from splitting set S by attribute A such that

$$S = \bigcup t \epsilon T \, t$$

- p(t) − The proportion of the number of elements in t to the number of elements in set S.
- H(t) − Entropy of subset t.

In ID3, information gain can be calculated (instead of entropy) for each remaining attribute. The attribute with the largest information gain is used to split the set S on that particular iteration.

What are the steps in ID3 algorithm?

The **steps in ID3 algorithm** are as follows:

1. Calculate entropy for dataset.
2. For each attribute/feature.
   2.1. Calculate entropy for all its categorical values.
   2.2. Calculate information gain for the feature.
3. Find the feature with maximum information gain.
4. Repeat it until we get the desired tree.

2.3 GINI IMPURITY – CLASSIFICATION ALGORITHM

DECISION TREES

Tree based methods – Decision Trees

→ Tree−based machine learning methods are among the most commonly used supervised learning methods. They are constructed by two entities; branches and nodes.

→ Tree−based ML methods are built by recursively splitting a training sample, using different features from a dataset at each node that splits the data most effectively.

→ The splitting is based on learning simple decision rules inferred from the training data. Generally, tree−based ML methods are simple and intuitive; to predict a class label or value, we start from the top of the tree or the root and, using branches, go to the nodes by comparing features on the basis of which will provide the best split.

→ Tree−based methods also use the mean for continuous variables or mode for categorical variables when making predictions on training observations in the regions they belong to. Since the set of rules used to segment the predictor space can be summarized in a visual representation with branches that show all the possible outcomes, these approaches are commonly referred to as decision tree methods.

→ The methods are flexible and can be applied to either classification or regression problems. *Classification and Regression Trees* (CART) is a commonly used term by Leo Breiman, referring to the flexibility of the methods in solving both linear and non−linear predictive modeling problems.

Types of Decision Trees

Decision trees can be classified based on the type of target or response variable.

i. Classification Trees

The default type of decision trees, used when the response variable is categorical—i.e. predicting whether a team will win or lose a game.

ii. Regression Trees

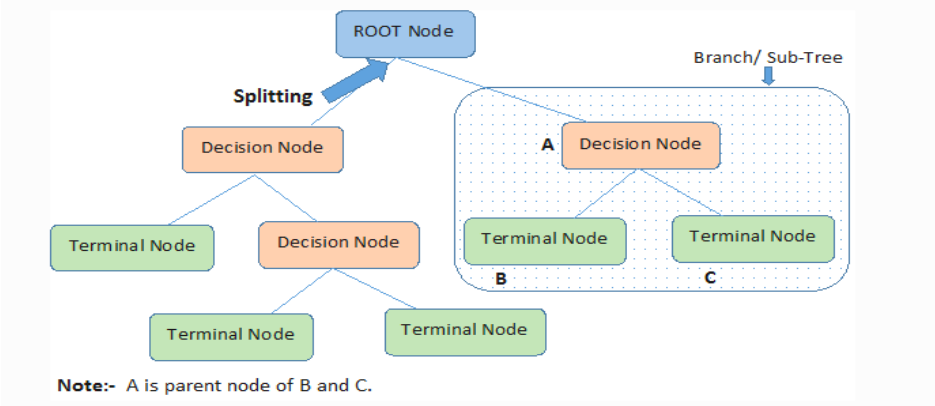
Used when the target variable is continuous or numerical in nature—i.e. predicting house prices based on year of construction, number of rooms, etc.

Advantages of Tree-based Machine Learning Methods

1. Interpretability: Decision tree methods are easy to understand even for non− technical people.
2. The data type isn't a constraint, as the methods can handle both categorical and numerical variables.
3. Data exploration

Disadvantages of Tree-based Machine Learning Methods

1. Large decision trees are complex, time−consuming and less accurate in predicting outcomes.
2. Decision trees don't fit well for continuous variables, as they lose important information when segmenting the data into different regions.



i) Root node **—** this represents the entire population or the sample, which gets divided into two or more homogenous subsets.

ii) Splitting **—** subdividing a node into two or more sub−nodes.

iii) ***Decision node*** **-** this is when a sub−node is divided into further sub−nodes.

iv) Leaf/Terminal node **-**this is the final/last node that we consider for our model output. It cannot be split further.

v) Pruning **—** removing unnecessary sub−nodes of a decision node to combat overfitting.

vi) Branch/Sub-tree **—** the sub−section of the entire tree.

vii) Parent and Child node **—** a node that's subdivided into a sub−node is a parent, while the sub−node is the child node.

CART (Classification and Regression Tree (CART)

CART (Classification And Regression Tree) is a variation of the decision tree algorithm. It can handle both classification and regression tasks. Scikit−Learn uses the Classification And Regression Tree (CART) algorithm to train Decision Trees (also called "growing" trees). CART was first produced by Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone in 1984.

CART Algorithm

→ CART is a predictive algorithm used in Machine learning and it explains how the target variable's values can be predicted based on other matters.

→ It is a decision tree where each fork is split into a predictor variable and each node has a prediction for the target variable at the end.

→ In the decision tree, nodes are split into sub−nodes on the basis of a threshold value of an attribute.

→ The root node is taken as the training set and is split into two by considering the best attribute and threshold value.

→ Further, the subsets are also split using the same logic. This continues till the last pure sub−set is found in the tree or the maximum number of leaves possible in that growing tree.

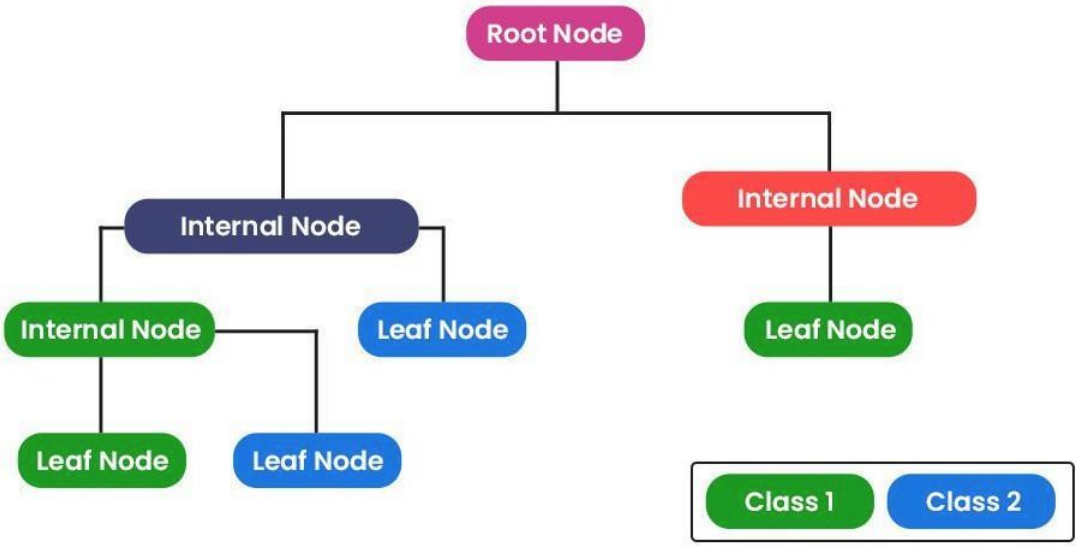
The CART algorithm works via the following process:

→ The best split point of each input is obtained.

→ Based on the best split points of each input in Step 1, the new "best" split point is identified.

→ Split the chosen input according to the "best" split point.

→ Continue splitting until a stopping rule is satisfied or no further desirable splitting is available.



CART algorithm uses Gini Impurity to split the dataset into a decision tree .It does that by searching for the best homogeneity for the sub nodes, with the help of the Gini index criterion.

Gini index/Gini impurity

→ The Gini index is a metric for the classification tasks in CART.

→ It stores the sum of squared probabilities of each class.

→ It computes the degree of probability of a specific variable that is wrongly being classified when chosen randomly and a variation of the Gini coefficient.

→ It works on categorical variables, provides outcomes either "successful" or "failure" and hence conducts binary splitting only.

The degree of the Gini index varies from 0 to 1,

→ Where 0 depicts that all the elements are allied to a certain class, or only one class exists there.

→ The Gini index of value 1 signifies that all the elements are randomly distributed across various classes &

→ A value of 0.5 denotes the elements are uniformly distributed into some classes.

Mathematically, we can write Gini Impurity as follows:

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

where pi is the probability of an object being classified to a particular class.

Classification tree

A classification tree is an algorithm where the target variable is categorical. The algorithm is then used to identify the "Class" within which the target variable is most likely to fall. Classification trees are used when the dataset needs to be split into classes that belong to the response variable (like yes or no).

Regression tree

A Regression tree is an algorithm where the target variable is continuous and the tree is used to predict its value. Regression trees are used when the response variable is continuous. For example, if the response variable is the temperature of the day.

CART model representation

CART models are formed by picking input variables and evaluating split points on those variables until an appropriate tree is produced.

Steps to create a Decision Tree using the CART algorithm:

→ **Greedy algorithm**: In this, The input space is divided using the Greedy method which is known as a recursive binary spitting. This is a numerical method within which all of the values are aligned and several other split points are tried and assessed using a cost function.

→ **Stopping Criterion:** As it works its way down the tree with the training data, the recursive binary splitting method described above must know when to stop splitting. The most frequent halting method is to utilize a minimum amount of training data allocated to every leaf node. If the count is smaller than the specified threshold, the split is rejected and also the node is considered the last leaf node.

→ **Tree pruning:** Decision tree's complexity is defined as the number of splits in the tree. Trees with fewer branches are recommended as they are simple to grasp and less prone to cluster the data. Working through each leaf node in the tree and evaluating the effect of deleting it using a hold−out test set is the quickest and simplest pruning approach.

→ **Data preparation for the CART:** No special data preparation is required for the CART algorithm.

Advantages of CART

→ Results are simplistic.

→ Classification and regression trees are Nonparametric and Nonlinear.

→ Classification and regression trees implicitly perform feature selection.

→ Outliers have no meaningful effect on CART.

→ It requires minimal supervision and produces easy−to−understand models.

Limitations of CART

→ Overfitting.

→ High Variance.

→ low bias.

→ the tree structure may be unstable.

Applications of the CART algorithm

→ For quick Data insights.

→ In Blood Donors Classification.

→ For environmental and ecological data.

→ In the financial sectors.

2.4 RULE BASED CLASSIFICATION

→ A Decision Tree in machine learning is a supervised learning algorithm that can be used for both classification and regression tasks.

→ In the context of classification, it functions as a rule-based system, creating a model that predicts the class of a target variable by learning simple decision rules inferred from the data features.

How it works as a Rule-Based Classifier:

Tree Structure:

A decision tree has a hierarchical, flowchart-like structure consisting of:

→ **Root Node:** The starting point, representing the entire dataset and the initial decision to be made.

→ **Internal Nodes:** Represent a test on an attribute or feature.

→ **Branches:** Represent the outcomes of the attribute test at an internal node.

→ **Leaf Nodes:** Represent the final decision or classification (the class label).

→ **Splitting Process:**

The algorithm recursively partitions the data based on the values of different features. At each internal node, a feature is selected that best splits the data into more homogeneous subsets with respect to the target variable. This selection is often based on metrics like Gini impurity or information gain.

→ **Rule Generation:**

Each path from the root node to a leaf node represents a set of decision rules. These rules are effectively "if-then-else" statements that define the conditions for classifying an instance into a particular class.

- For example, a rule might be: IF (Temperature > 25°C) AND (Humidity < 60%) THEN Class = "Play Tennis".

→ **Classification:**

To classify a new, unseen data point, it traverses the tree from the root node, following the branches corresponding to its feature values, until it reaches a leaf node. The class label associated with that leaf node is then assigned as the prediction for the new data point.

Key Aspects:

→ **Interpretability:** Decision trees are highly interpretable because the decision rules are explicit and can be easily understood by humans.

→ **Non-parametric:** They do not make assumptions about the underlying data distribution.

→ **Handling of various data types:** They can handle both numerical and categorical data.

→ **Overfitting:** Deep trees can overfit the training data, leading to poor generalization on unseen data. Techniques like pruning are used to mitigate this.

---

3. NAÏVE BAYESIAN CLASSIFICATION

3.1 NAÏVE BAYES ALGORITHM

→ Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.

→ It is mainly used in *text classification* that includes a high−dimensional training dataset.

→ Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

→ It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

→ Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

→ Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

→ The formula for Bayes' theorem is given as:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

where,

P(A/B) is Posterior probability*:* Probability of hypothesis A on the observed event B.

P(B/A) is Likelihood probability*:* Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood ***with the help of the below example:***

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So, to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the below dataset:

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying Bayes'theorem:

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= 0.35 P(Yes)=0.71

So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5 P(No)= 0.29

P(Sunny)= 0.35

So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

3.2 BAYESIAN BELIEF NETWORKS

→ Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty.

→ We can define a Bayesian network as: "A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph."

→ It is also called a **Bayes network, belief network, decision network**, or **Bayesian model**. Bayesian networks are probabilistic, because these networks are built from a **probability distribution**, and also use probability theory for prediction and anomaly detection.

→ Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network.

→ It can also be used in various tasks including **prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction**, and **decision making under uncertainty**.

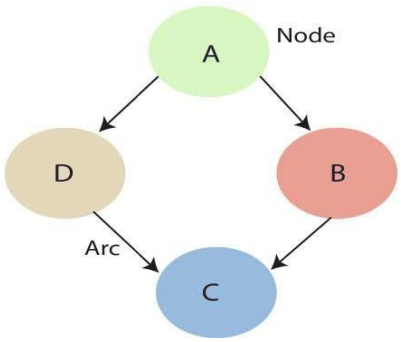
Bayesian Network can be used for building models from data and experts opinions, and it consists of two parts:

→ **Directed Acyclic Graph**

→ **Table of conditional probabilities.**

The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an **Influence diagram**.

A Bayesian network graph is made up of nodes and Arcs (directed links), where:



→ Each **node** corresponds to the random variables, and a variable can be **continuous** or **discrete**.

→ **Arc or directed arrows** represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.

→ These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other.

→ *In the above diagram, A, B, C, and D are random variables represented by the nodes of the network graph.*

→ *If we are considering node B, which is connected with node A by a directed arrow, then node A is called the parent of Node B.*

→ *Node C is independent of node A.*

The Bayesian network has mainly two components:

- **Causal Component**
- **Actual numbers**

Each node in the Bayesian network has condition probability distribution **P(Xi |Parent(Xi))**, which determines the effect of the parent on that node. Bayesian network is based on Joint probability distribution and conditional probability. So

let's first understand the joint probability distribution:

Joint probability distribution:

If we have variables x1, x2, x3,. ..., xn, then the probabilities of a different combination of x1, x2, x3.. xn, are known as Joint probability distribution.

P[x1, x2, x3,.., xn], it can be written as the following way in terms of the joint probability distribution.

= P[x1| x2, x3,....., xn]P[x2, x3,.....,xn]

= P[x1| x2, x3,....., xn]P[x2|x3,....., xn] P[xn-1|xn]P[xn].

In general for each variable Xi, we can write the equation as:

$$P(X_i|X_{i-1},\ldots,X_1) = P(X_i \mid Parents(X_i))$$

Problem:

Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called the Harry.

Solution:

→ The Bayesian network for the above problem is given below. The network structure is showing that burglary and earthquake is the parent node of the alarm and directly affecting the probability of alarm's going off, but David and Sophia's calls depend on alarm probability.

→ The network is representing that our assumptions do not directly perceive the burglary and also do not notice the minor earthquake, and they also not confer before calling.

→ The conditional distributions for each node are given as conditional probabilities table or CPT.

→ Each row in the CPT must be sum to 1 because all the entries in the table represent an exhaustive set of cases for the variable.

→ In CPT, a boolean variable with k boolean parents contains 2^K probabilities. Hence, if there are two parents, then CPT will contain 4 probability values

List of all events occurring in this network:

→ *Burglary (B)*

→ *Earthquake(E)*

→ *Alarm(A)*

→ *David Calls(D)*

→ *Sophia calls(S)*

We can write the events of problem statement in the form of probability: **P[D, S, A, B, E]**, can rewrite the above probability statement using joint probability distribution:

P[D, S, A, B, E]= P[D | S, A, B, E]. P[S, A, B, E]

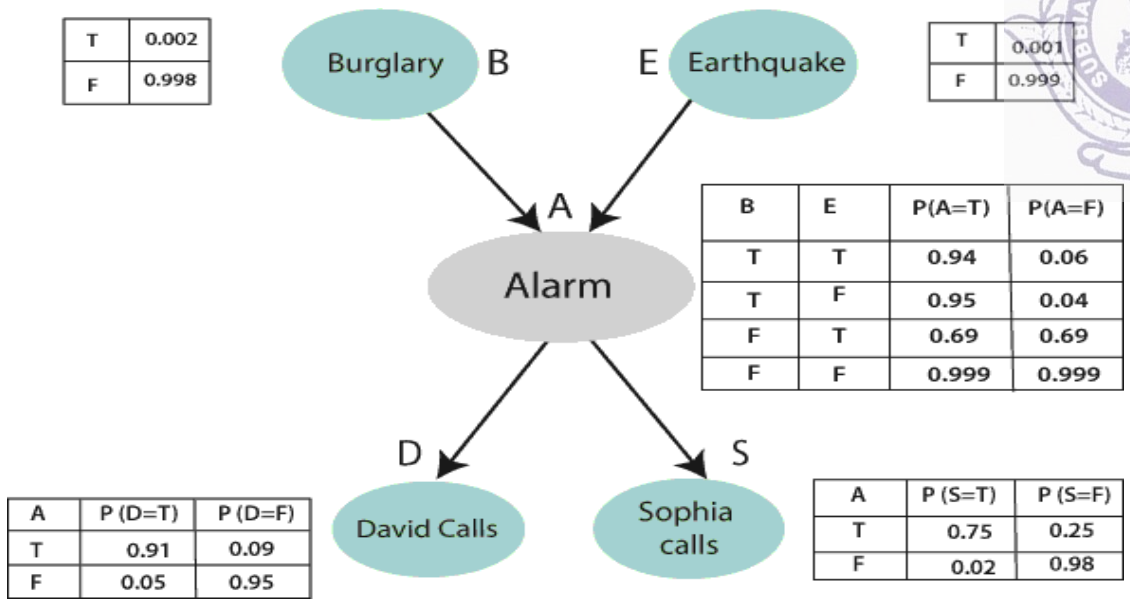
=P[D | S, A, B, E]. P[S | A, B, E]. P[A, B, E]

= P [D| A]. P [S| A, B, E]. P[A, B, E]

= P[D | A]. P[S | A]. P[A| B, E]. P[B, E]

= P[D | A]. P[S | A]. P[A| B, E]. P[B |E]. P[E]

Let's take the observed probability for the Burglary and earthquake component: P(B= True) = 0.002, which is the probability of burglary.



T	0.002
F	0.998

T	0.001
F	0.999

B	E	P(A=T)	P(A=F)
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.69	0.69
F	F	0.999	0.999

A	P (D=T)	P (D=F)
T	0.91	0.09
F	0.05	0.95

A	P (S=T)	P (S=F)
T	0.75	0.25
F	0.02	0.98

P(B= False)= 0.998, which is the probability of no burglary.

P(E= True)= 0.001, which is the probability of a minor earthquake

P(E= False)= 0.999, Which is the probability that an earthquake not occurred.

We can provide the conditional probabilities as per the below tables:

Conditional probability table for Alarm A:

The Conditional probability of Alarm A depends on Burglar and earthquake:

B	E	P(A= True)	P(A= False)
True	True	0.94	0.06
True	False	0.95	0.04
False	True	0.31	0.69
False	False	0.001	0.999

Conditional probability table for David Calls:

The Conditional probability of David that he will call depends on the probability of Alarm.

A	P(D= True)	P(D= False)
True	0.91	0.09
False	0.05	0.95

Conditional probability table for Sophia Calls:

The Conditional probability of Sophia that she calls is depending on its Parent Node "Alarm."

A	P(S= True)	P(S= False)
True	0.75	0.25
False	0.02	0.98

From the formula of joint distribution, we can write the problem statement in the form of probability distribution:

P(S, D, A, ¬B, ¬E) = P (S|A) *P (D|A)*P (A|¬B ^ ¬E) *P (¬B) *P (¬E).

= 0.75* 0.91* 0.001* 0.998*0.999

= 0.00068045.

Hence, a Bayesian network can answer any query about the domain by using Joint distribution.

The semantics of Bayesian Network:

There are two ways to understand the semantics of the Bayesian network, which is given below:

1) **To understand the network as the representation of the Joint probability distribution.** It is helpful to understand how to construct the network.
2) **To understand the network as an encoding of a collection of conditional independence statements.**

---

4. Support Vector Machines (SVM)

→ Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.

→ However, primarily, it is used for Classification problems in Machine Learning.

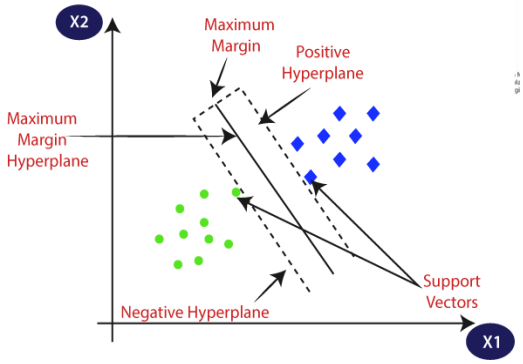
→ The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n−dimensional space into classes so that we can easily put the new data point in the correct category in the future.

→ This best decision boundary is called a **hyperplane.**

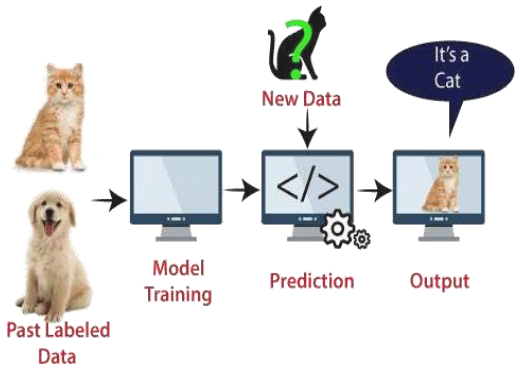
→ SVM chooses the extreme points/vectors that help in creating the hyperplane.

→ These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

→ Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature.



So, as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:

SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

Types of SVM

SVM can be of two types:

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

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

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n−dimensional space, but we need to find out the best decision boundary that helps to classify the data points.

This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2−dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

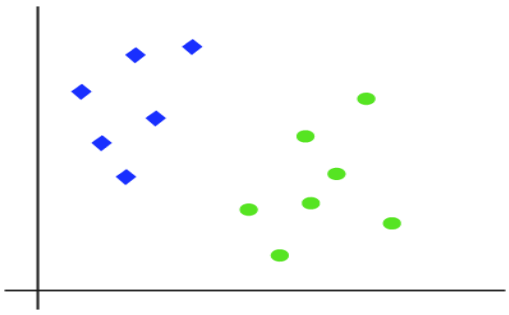
Support Vectors:

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

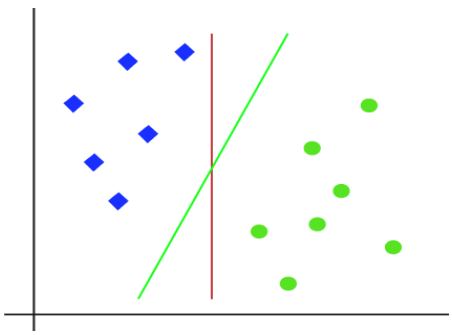
How does SVM works?

Linear SVM:

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair (x1, x2) of coordinates in either green or blue. Consider the below image:



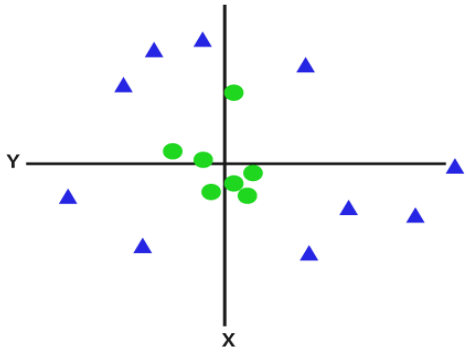
So, as it is 2−d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

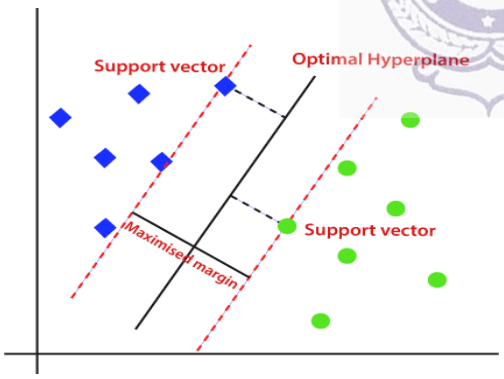


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane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

Non-Linear SVM:

If data is linearly arranged, then we can separate it by using a straight line, but for non−linear data, we cannot draw a single straight line. Consider the below image:

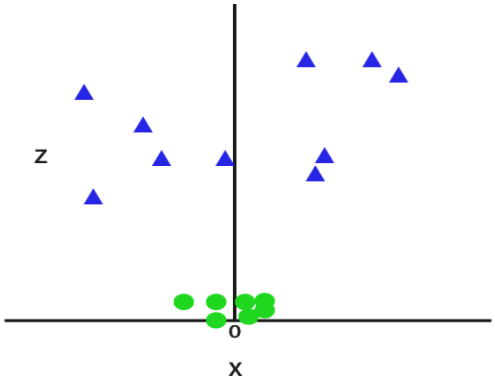




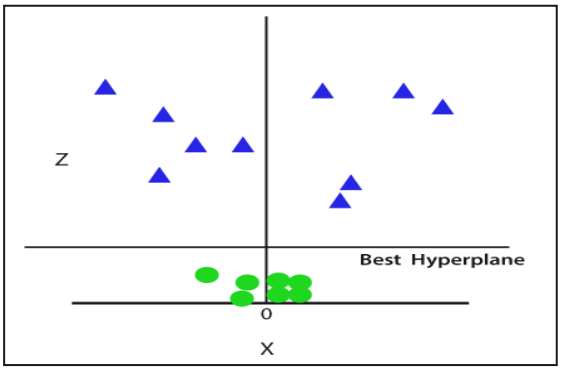
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non−linear data, we will add a third-dimension z. It can be calculated as:

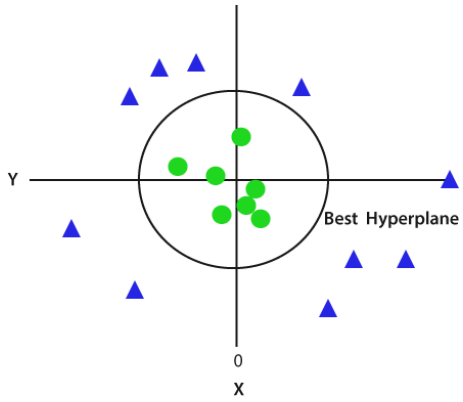
$$z=x^2+y^2$$

By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:





Hence, we get a circumference of radius 1 in case of non−linear data.

Kernel Methods in SVM

→ **Kernel Function** is a method used to take data as input and transform it into the required form of processing data.

→ "Kernel" is used due to a set of mathematical functions used in Support Vector Machine providing the window to manipulate the data. So, Kernel Function generally transforms the training set of data so that a non−linear decision surface is able to transform to a linear equation in a higher number of dimension spaces.

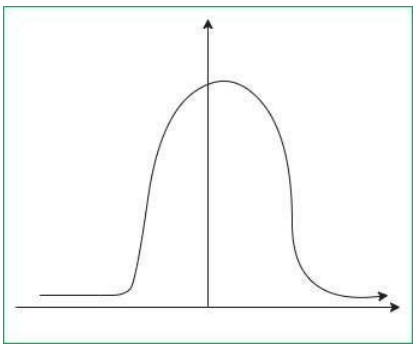
→ Basically, It returns the inner product between two points in a standard feature dimension.

Major Kernel Functions:-

For Implementing Kernel Functions, first of all, we have to install the "scikit−learn" library using the command prompt terminal:

pip install scikit−learn

- **Gaussian Kernel:** It is used to perform transformation when there is no prior knowledge about data.
- **Gaussian Kernel Radial Basis Function (RBF):** Same as above kernel function, adding radial basis method to improve the transformation.



OCS351 – ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FUNDAMENTALS

UNIT V UNSUPERVISED LEARNING

Unsupervised Learning – Principal Component Analysis – **Neural Network:** Fixed weight competitive Nets – Kohonen Self-Organizing Feature Maps – **Clustering:** Definition – Types of clustering – Hierarchical clustering algorithm – k-means algorithm

1. Unsupervised Learning

→ **Unsupervised learning:** This type of machine learning involves algorithms that train on unlabeled data.

→ The algorithm scans through data sets looking for any meaningful connection.

→ The data that algorithms train on as well as the predictions or recommendations they output are predetermined.

How does unsupervised machine learning work?

→ Unsupervised machine learning algorithms do not require data to be labeled.

→ They sift through unlabeled data to look for patterns that can be used to group data points into subsets.

→ Most types of deep learning, including neural networks, are unsupervised algorithms.

→ Unsupervised learning algorithms are good for the following tasks:

- **Clustering:** Splitting the dataset into groups based on similarity.
- **Anomaly detection:** Identifying unusual data points in a data set.
- **Association mining:** Identifying sets of items in a data set that frequently occur together.
- **Dimensionality reduction:** Reducing the number of variables in a data set.

Unsupervised Learning - Users have to look at the data and then divide it based on its own algorithms without having any training. There is no target or outcome variable to predict nor estimate.

1.1 PRINCIPAL COMPONENT ANALYSIS

→ Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning.

→ It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation.

→ These new transformed features are called the **Principal Components**.

→ It is one of the popular tools that is used for exploratory data analysis and predictive modeling.

→ It is a technique to draw strong patterns from the given dataset by reducing the variances.

→ PCA generally tries to find the lower−dimensional surface to project the high− dimensional data.

→ PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality.

→ Some real−world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels.***

→ It is a feature extraction technique, so it contains the important variables and drops the least important variable.

The PCA algorithm is based on some mathematical concepts such as:

→ Variance and Covariance

→ Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

→ **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.

→ **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from −1 to +1. Here, −1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.

→ **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.

→ **Eigenvectors:** If there is a square matrix M, and a non−zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.

→ **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Principal Components in PCA

As described above, the transformed new features or the output of PCA are the Principal Components. The number of these PCs are either equal to or less than the original features present in the dataset. Some properties of these principal components are given below:

→ The principal component must be the linear combination of the original features.

→ These components are orthogonal, i.e., the correlation between a pair of variables is zero.

→ The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will have the least importance.

Steps for PCA algorithm

1. **Getting the dataset -** Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.

2. **Representing data into a structure -** Now we will represent our dataset into a structure. Such as we will represent the two−dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.
3. **Standardizing the data -** In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance.

If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.

4. **Calculating the Covariance of Z -** To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.
5. **Calculating the Eigen Values and Eigen Vectors -** Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.
6. **Sorting the Eigen Vectors -** In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P*.
7. **Calculating the new features Or Principal Components -** Here we will calculate the new features. To do this, we will multiply the P* matrix to the Z. In the resultant matrix Z*, each observation is the linear combination of original features. Each column of the Z* matrix is independent of each other.
8. **Remove less or unimportant features from the new dataset -** The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out.

Applications of Principal Component Analysis

→ PCA is mainly used as the dimensionality reduction technique in various AI applications such **as computer vision, image compression, etc.**

→ It can also be used for finding hidden patterns if data has high dimensions. Some fields where PCA is used are Finance, data mining, Psychology, etc.

2. NEURAL NETWORK:

2.1 FIXED WEIGHT COMPETITIVE NETS

Fixed weight competitive Nets:

→ During training process also the weights remains fixed in these competitive networks.

→ The idea of competition is used among neurons for enhancement of contrast in their activation functions.

→ In this, two networks- **Maxnet and Hamming networks.**

Maxnet

→ Maxnet network was developed by Lippmann in 1987.

→ The maxnet serves as a sub net for picking the node whose input is larger.

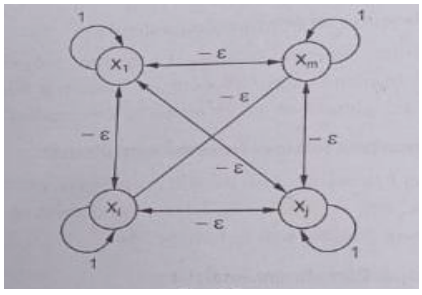
→ All the nodes present in this subnet are fully interconnected and there exist symmetrical weights in all these weighted interconnections.

Architecture of Maxnet

→ The architecture of Maxnet is a fixed symmetrical weights are present over the weighted interconnections.

→ The weights between the neurons are inhibitory and fixed.

→ The Maxnet with this structure can be used as a subnet to select a particular node whose net input is the largest.



Testing Algorithm of Maxnet

The Marker uses the following activation function:

$$f(x) = \begin{cases} x & if\ x > 0 \\ 0 & if\ z \leq 0 \end{cases}$$

Testing Algorithm

Step 0: Initial weights and initial activations are set. The weight is set as $[0 < \varepsilon < 1/m]$, where 'm' is the total number of nodes. Let

$$X_j\ (0) = \text{Input the node } X_j$$

and

$$w_{ij} = \begin{cases} i & if\ i > j \\ -\varepsilon & if\ i \neq j \end{cases}$$

Step 1: Perform steps 2-4, when stopping condition is false

Step 2: Update the activations of each node. For $j = 1$ to m,

$$X_j\ (\text{new}) = F\ [\ X_j\ (\text{old}) - \varepsilon \sum_{i \neq j} X_k\ (\text{old})]$$

Step 3: Save the activations of each node. For $j = 1$ to m,

$$X_j\ (\text{new}) = X_j\ (\text{old})$$

Step 4: Finally, test the stopping condition for convergence of the network. The following is the stopping condition; If more than one node has a nonzero activation continue; else stop

Hamming Network

→ The Hamming network is a two-layer feed forward neural network for classification of binary bipolar n-tuple input vectors using minimum Hamming distance denoted as D_H (Lippmann, 1987).

→ The first layer is the input layer for the n-tuple input vectors.

→ The second layer (also called the memory layer) stores p memory patterns.

→ A p-class Hamming network has p output neurons in this layer. The strongest response of a neuron is indicative of the minimum Hamming distance between the stored pattern and the input vector.

Hamming Distance

Hamming distance of two vectors, x and y dimension n

$$x.\ y = a - \dot{a}$$

where: a is number of bits in agreement in x and y (No. of similarities bits in x and y), and d is number of bits different in x and y (No. of dissimilarities bits x and y).

The value *"a – d"* is the Hamming distance existing between two vectors. Since, the total number of components is n, we have

$$n = a + d$$

i.e $$d = n - a$$

On Simplification, we get

$$x.\ y = a - (n - a)$$

$$x.\ y = 2a - n$$

$$2a = x.\ y + n$$

$$a = \frac{1}{2} x.\ y + \frac{1}{2}\ n$$

From the above equation, it is clearly understood that the weights can be set to one-half the exemplar vector and bias can be set initially to n/2.

Testing Algorithm of Hamming Network

Step 0: Initialize the weights for $i = 1$ to n, and $j = 1$ to m.

$$W_{ij} = \frac{e_t(j)}{2}$$

Initialize the bias for storing the 'm' exemplar vectors. For $j = 1$ to m,

$$b_j = \frac{n}{x}$$

Step 1: Perform steps 2 – 4 for each input vector x.

Step 2: Calculate the net input to each unit Y_j, i.e.,

$$y_{inj} = \sum_{i=1}^{n} x_i w_{ij} + b_j = 1 \text{ to } m$$

Step 3: Initialize the activations for Maxnet, i.e.,

$$y_j(0) = y_{inj} \; j = 1 \text{ to } m$$

Step 4: Maxnet is found to iterate for finding the exemplar that best matches the input patterns.

2.2 KOHONEN SELF-ORGANIZING FEATURE MAPS

Kohonen Self-Organizing Feature Maps

→ Kohonen Self-Organizing feature map (SOM) refers to a neural network, which is trained using competitive learning. Basic competitive learning implies that the competition process takes place before the cycle of learning.

→ The competition process suggests that some criteria select a winning processing element. After the winning processing element is selected, its weight vector is adjusted according to the used learning law (Hecht Nielsen 1990).

→ The Self-organizing map makes topologically ordered mappings between input data and processing elements of the map.

→ Topological order implies that if two inputs are of similar characteristics, the most active processing elements answering to inputs that are located closed to each other on the map.

→ The weight vectors of the processing elements are organized in ascending to descending order.

→ $W_i < W_{i+1}$ for all values of i or W_{i+1} for all values of i (this definition is valid for one-dimensional self-organizing map only).

→ The self-organizing map is typically represented as a two-dimensional sheet of processing elements described in the figure given below.

→ Each processing element has its own weight vector, and learning of SOM (self-organizing map) depends on the adaptation of these vectors.

→ The processing elements of the network are made competitive in a self-organizing process, and specific criteria pick the winning processing element whose weights are updated.

→ Generally, these criteria are used to limit the Euclidean distance between the input vector and the weight vector.

→ SOM (self-organizing map) varies from basic competitive learning so that instead of adjusting only the weight vector if the winning processing element also weight vectors of neighboring processing elements are adjusted.

→ First, the size of the neighborhood is largely making the rough ordering of SOM and size is diminished as time goes on.

→ At last, only a winning processing element is adjusted, making the fine-tuning of SOM possible.

→ The use of neighborhood makes topologically ordering procedure possible and together with competitive learning makes the process non-linear.

→ It was discovered by Finish professor and researcher Dr. TeuvoKohonen in 1982.

→ The self-organizing map refers to an unsupervised learning model proposed for applications in which maintaining a topology between input and output spaces.

→ The notable attribute of this algorithm is that the input vectors that are close and similar in high dimensional space are also mapped to close by nodes in the 2D space.

→ It is fundamentally a method for dimensionality reduction, as it maps high-dimension inputs to a low dimensional discrete representation and preserves the basic structure of its input space.

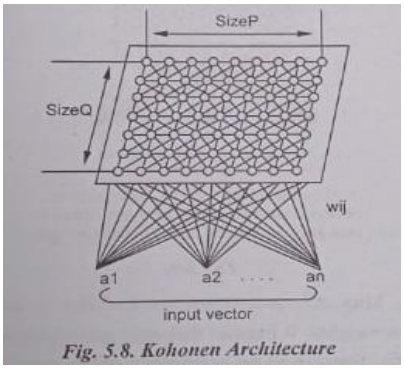


Fig. 5.8. Kohonen Architecture

→ All the entire learning process occurs without supervision because the nodes are self-organizing.

→ They are also known as feature maps, as they are basically retraining the features of the input data, and simply grouping themselves as indicated by the similarity between each other.

→ It has practical value for visualizing complex or huge quantities of high dimensional data and showing the relationship between them into a low, usually two-dimensional field to check whether the given unlabeled data have any structure to it.

→ A self-organizing Map (SOM) varies from typical artificial neural networks (ANNs) both in its architecture and algorithmic properties.

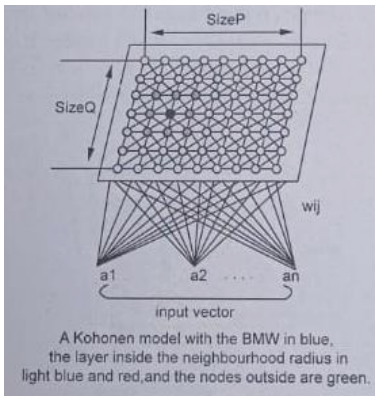
→ Its structure consists of a single layer linear 2D grid of neurons, rather than a series of layers.

→ All the nodes on this lattice are associated directly to the input vector, but not to each other.

→ It means the nodes don't know the values of their neighbors, and only update the weight of their associations as a function of the given input.

→ The grid itself is the map that coordinates itself at each iteration as a function of the input data.

→ As such, after clustering, each node has its own coordinate (i, j), which enables one to calculate Euclidean distance between two nodes by means of the Pythagoras theorem.



A Kohonen model with the BMW in blue, the layer inside the neighbourhood radius in light blue and red,and the nodes outside are green.

→ A self-organizing Map utilizes competitive learning instead of error-correction learning to modify its weights.

→ It implies that only an individual node is activated at each cycle in which the features of an occurrence of the input vector are introduced to the neural network, as all nodes compete for the privilege to respond to the input.

→ The selected node-the Best Matching Unit (BMU) is selected according to the similarity between the current input values and all the other nodes in the network.

→ The node with the fractional Euclidean difference between the input vector, all nodes, and its neighboring nodes is selected and within a specific radius, to have their position slightly adjusted to coordinate the input vector.

→ By experiencing all the nodes present on the grid, the whole grid eventually matches the entire input dataset with connected nodes gathered towards one area, and dissimilar ones are isolated.

Algorithm:

Step 1: Each node weight *w_ij* initialize to a random value.

Step 2: Choose a random input vector *x_k.*

Step 3: Repeat steps 4 and 5 for all nodes on the map.

Step 4: Calculate the Euclidean distance between weight vector w_{ij} and the input vector *x(t)* connected with the first node, where $t, i, j = 0$.

Step 5: Track the node that generates the smallest distance *t*.

Step 6: Calculate the overall Best Matching Unit (BMU). It means the node with the smallest distance from all calculated ones.

Step 7: Discover topological neighborhood $\beta_{ij}(t)$ its radius $\sigma(t)$ of BMU in Kohonen Map.

Step 8: Repeat for all nodes in the BMU neighborhood: Update the weight vector *w_ij* of the first node in the neighborhood of the BMU by including a fraction of the difference between the input vector *x(t)* and the weight *w(t)* of the neuron.

Step 9: Repeat the complete iteration until reaching the selected iteration limit $t = n$.

Here, step 1 represents initialization phase, while steps 2 to 9 represents the training phase.

where; t = current iteration.

i = row coordinate of the nodes grid.

J = column coordinate of the nodes grid.

W = weight vector

w_ij = association weight between the nodes i, j in the grid.

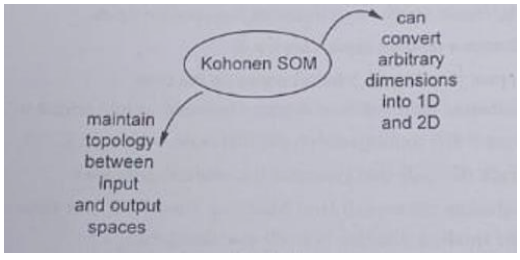
X = input vector.

X(t) = the input vector instance at iteration t.

β_ij = the neighborhood function, decreasing and representing node i, j distance from the BMU.

$\sigma(t)$ = The radius of the neighborhood function, which calculates how far neighbor nodes are examined in the 2D grid when updating vectors. It gradually decreases over time.

A **Kohonen Self-Organizing Map** is a map that is used for maintaining neighbor topology. It is referred to as a neural network that is trained by competitive learning.



Neighbor topologies

Below are the two most used topologies in the Kohonen Self-Organizing Map.

Rectangular grid

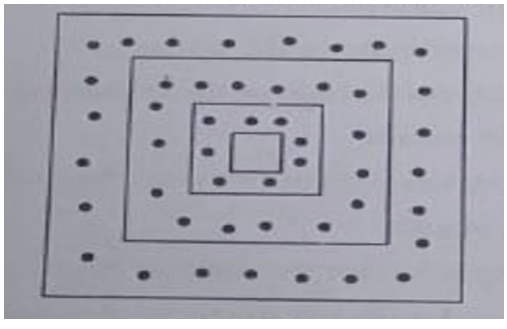
The following explains the **rectangular grid topology:**

→ In the distance of 2 grids; 24 nodes

→ In the distance of 1 grid; 16 nodes

→ In the distance of 0 grid; 8 nodes

There is a total difference of 88 nodes between each grid.



Hexagonal Topology

The following explains the hexagonal grid topology.

→ In the distance of 2 grids: 18 nodes

→ In the distance of 1 grid; 12 nodes

→ In the distance of 0 grid; 6 nodes

There is a total difference of 8 nodes between each grid.

Each node has its own coordinates after clustering, which can be calculated with the Pythagorean theorem using the Euclidean distance between the two nodes.

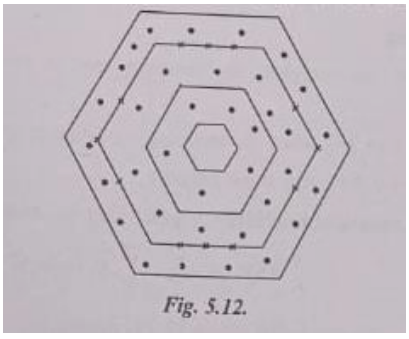


Fig. 5.12.

Structure

A Kohonen Self-Organizing Map is different from common artificial neural netwotks in its:

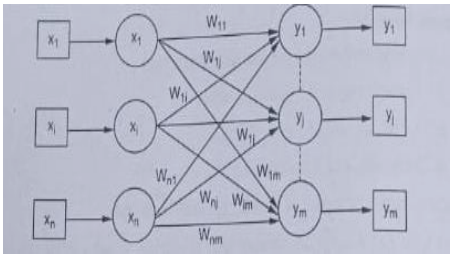
→ Architecture properties

→ Algorithmic properties

A kohonen self-organizing map consists of a single layer 2D grid of neurons. The nodes do not know the values of their neighbors. The weights of links are updated as a function of the given inputs. However, all the nodes on the grid are directly linked to the input vectors.

Architecture

The architecture of KSOM is similar to that of the competitive network. With the help of neighbourhood schemas, discussed earlier, the training can take place over the extended region of the network.



Algorithm for training

Step 1: Initialize the weights, the learning rate α and the neighbourhood topological scheme.

Step 2: Continue step 3-9, when the stopping condition is not true.

Step 3: Continue 4-6 for every input vector x.

Step 4: Calculate square of Euclidean distance for $j = 1$ to m

$$D(j) = \sum_{i=1}^{n} \sum_{j=1}^{m} (x_i - w_{ij})^2$$

Step 5: Obtain the winning unit J where D j is minimum.

Step 6: Calculate the new weight of the winning unit by the following relation –

$$w_{ij}\ (\text{new}) = w_{ij}\ (\text{old}) + \alpha\ [x_i - w_{ij}(\text{old})]$$

Step 7: Update the learning rate α by the following relation-

$$\alpha(t+1) = 0.5\ \alpha t$$

Step 8: Reduce the radius of topological scheme.

Step 9: Check for the stopping condition for the network.

---

3. CLUSTERING:

3.1 DEFINITION & TYPES OF CLUSTERING

→ Clustering is an unsupervised machine learning task. You might also hear this referred to as cluster analysis because of the way this method works.

→ Using a clustering algorithm means you're going to give the algorithm a lot of input data with no labels and let it find any groupings in the data it can. Those groupings are called *clusters*.

→ A cluster is a group of data points that are similar to each other based on their relation to surrounding data points.

→ Clustering is used for things like feature engineering or pattern discovery. When you're starting with data you know nothing about, clustering might be a good place to get some insight.

Types of clustering algorithms

There are different types of clustering algorithms that handle all kinds of unique data.

Density-based

→ In density−based clustering, data is grouped by areas of high concentrations of data points surrounded by areas of low concentrations of data points.

→ Basically the algorithm finds the places that are dense with data points and calls those clusters.

→ The great thing about this is that the clusters can be any shape. You aren't constrained to expected conditions.

→ The clustering algorithms under this type don't try to assign outliers to clusters, so they get ignored.

Distribution-based

→ With a distribution−based clustering approach, all of the data points are considered parts of a cluster based on the probability that they belong to a given cluster.

→ It works like this: there is a center−point, and as the distance of a data point from the center increases,

→ If you aren't sure of how the distribution in your data might be, you should consider a different type of algorithm.

Centroid-based

→ Centroid−based clustering is the one you probably hear about the most. It's a little sensitive to the initial parameters you give it, but it's fast and efficient.

→ These types of algorithms separate data points based on multiple centroids in the data. Each data point is assigned to a cluster based on its squared distance from the centroid. This is the most commonly used type of clustering.

Hierarchical-based

→ Hierarchical−based clustering is typically used on hierarchical data, like you would get from a company database or taxonomies.

→ It builds a tree of clusters so everything is organized from the top−down.

→ This is more restrictive than the other clustering types, but it's perfect for specific kinds of data sets.

When to use clustering

→ When you have a set of unlabeled data, it's very likely that you'll be using some kind of unsupervised learning algorithm.

→ There are a lot of different unsupervised learning techniques, like neural networks, reinforcement learning, and clustering.

→ The specific type of algorithm you want to use is going to depend on what your data looks like.

→ You might want to use clustering when you're trying to do anomaly detection to try and find outliers in your data.

→ It helps by finding those groups of clusters and showing the boundaries that would determine whether a data point is an outlier or not.

→ If you aren't sure of what features to use for your machine learning model, clustering discovers patterns you can use to figure out what stands out in the data.

3.3 HIERARCHICAL CLUSTERING ALGORITHM

Hierarchical clustering, also known as hierarchical cluster analysis or HCA, is another unsupervised machine learning approach for grouping unlabeled datasets into clusters.

The hierarchy of clusters is developed in the form of a tree in this technique, and this tree−shaped structure is known as the dendrogram.

Each observation is treated as a separate cluster in hierarchical clustering. After that, it repeats the next two steps:

1. Finds the two clusters that are the closest together.

2. Combines the two clusters that are the most similar. This iterative process is repeated until all of the clusters have been integrated.

Hierarchical clustering method functions in two approaches-

→ Agglomerative

→ Divisive

1. Agglomerative clustering:

Agglomerative Clustering is a bottom−up strategy in which each data point is originally a cluster of its own, and as one travels up the hierarchy, more pairs of clusters are combined. In it, two nearest clusters are taken and joined to form one single cluster.

The algorithm for Agglomerative Hierarchical Clustering is:

→ Calculate the similarity of one cluster with all the other clusters (calculate proximity matrix).

→ Consider every data point as an individual cluster

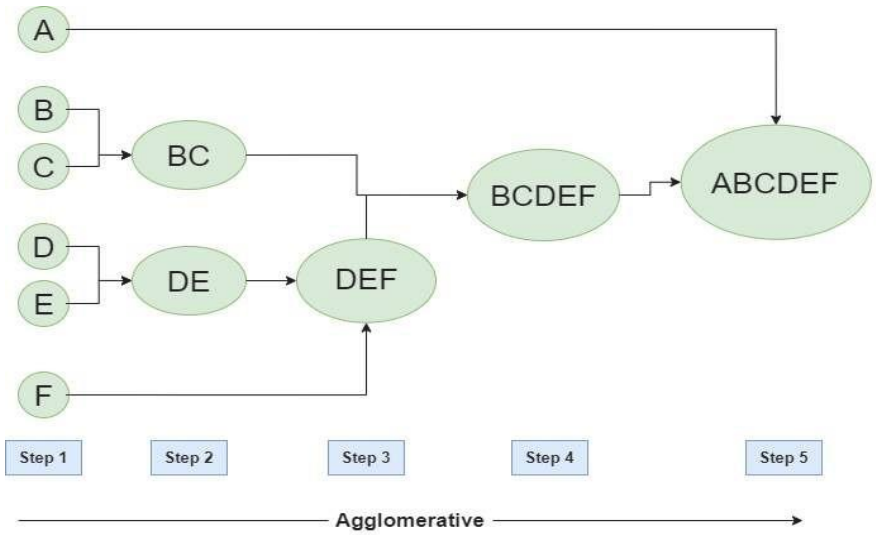
→ Merge the clusters which are highly similar or close to each other.

→ Recalculate the proximity matrix for each cluster

→ Repeat Steps 3 and 4 until only a single cluster remains.

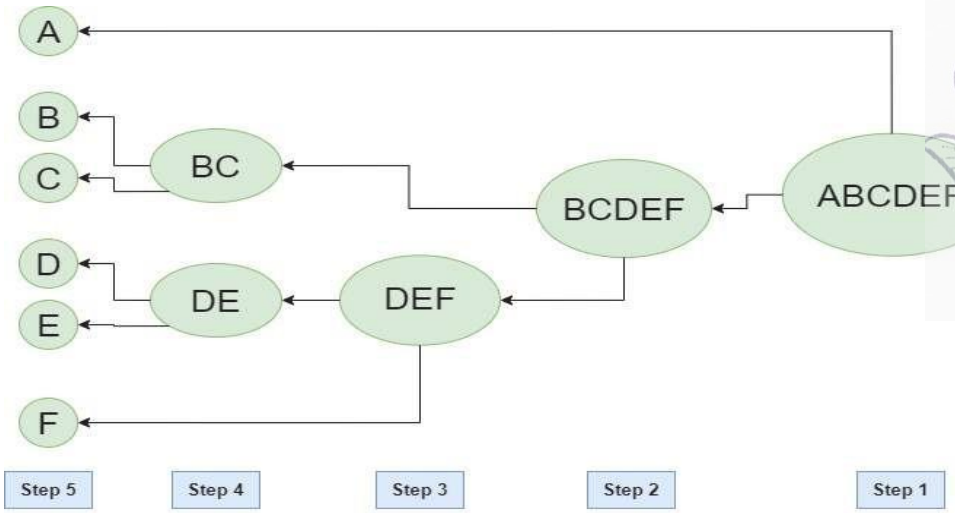
Note: This is just a demonstration of how the actual algorithm works no calculation has been performed below all the proximity among the clusters is assumed.

Let's say we have six data points A, B, C, D, E, and F.



2. Divisive clustering:

The divisive clustering algorithm is a top−down clustering strategy in which all points in the dataset are initially assigned to one cluster and then divided iteratively as one progresses down the hierarchy. It partitions data points that are clustered together into one cluster based on the slightest difference. This process continues till the desired number of clusters is obtained.



3.4 K-MEANS ALGORITHM

Every Machine Learning engineer wants to achieve accurate predictions with their algorithms. Such learning algorithms are generally broken down into two types − supervised and unsupervised. K−Means clustering is one of the unsupervised algorithms where the available input data does not have a labeled response.

→ K−means is a centroid−based clustering algorithm, where we calculate the distance between each data point and a centroid to assign it to a cluster. The goal is to identify the K number of groups in the dataset.

→ *"K−means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster."*

→ It is an iterative process of assigning each data point to the groups and slowly data points get clustered based on similar features. The objective is to minimize the sum of distances between the data points and the cluster centroid, to identify the correct group each data point should belong to.

→ Here, we divide a data space into K clusters and assign a mean value to each. The data points are placed in the clusters closest to the mean value of that cluster. There are several distance metrics available that can be used to calculate the distance.

How does K-means work?

Let's take an example to understand how K−means work step by step. The algorithm can be broken down into 4−5 steps.

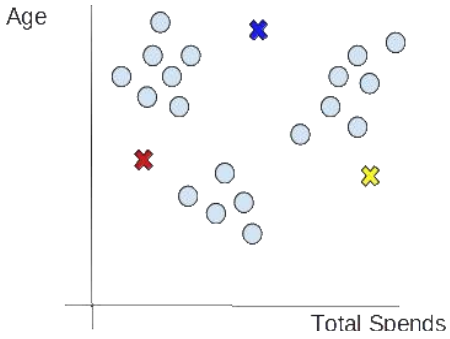
1. Choosing the number of clusters

The first step is to define the K number of clusters in which we will group the data.

Let's select K=3.

2. Initializing centroids

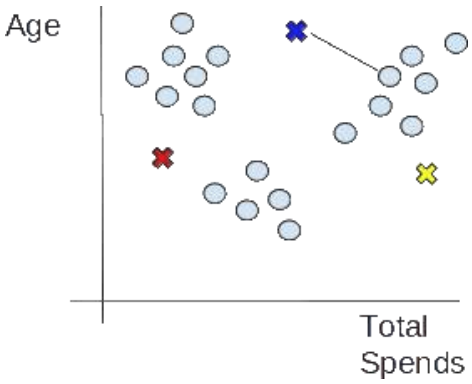
Centroid is the center of a cluster but initially, the exact center of data points will be unknown so, we select random data points and define them as centroids for each cluster. We will initialize 3 centroids in the dataset.



K−means clustering – centroid

3. Assign data points to the nearest cluster

Now that centroids are initialized, the next step is to assign data points X_n to their closest cluster centroid C_k.

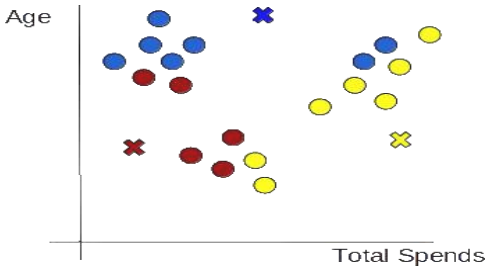


K−means clustering – assign data points

In this step, we will first calculate the distance between data point X and centroid C using Euclidean Distance metric.

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

And then choose the cluster for data points where the distance between the data point and the centroid is minimum.

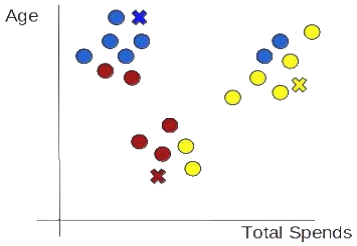


K−means clustering

4. Re-initialize centroids

Next, we will re−initialize the centroids by calculating the average of all data points of that cluster.

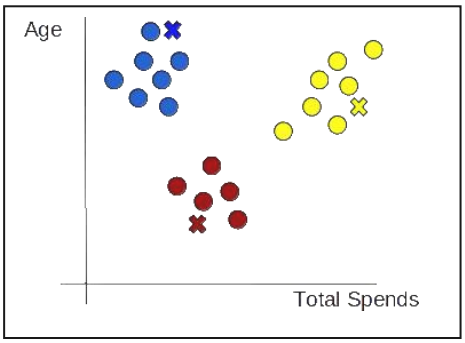
$$Ci = \frac{1}{|Ni|} \sum xi$$



K−means clustering

5. Repeat steps 3 and 4

We will keep repeating steps 3 and 4 until we have optimal centroids and the assignments of data points to correct clusters are not changing anymore. Figure − *K−means clustering.*



Does this iterative process sound familiar? Well, K−means follows the same approach as Expectation−Maximization(EM). EM is an iterative method to find the maximum likelihood of parameters where the machine learning model depends on unobserved features. This approach consists of two steps Expectation(E) and Maximization(M) and iterates between these two.

For K−means, The Expectation(E) step is where each data point is assigned to the most likely cluster and the Maximization(M) step is where the centroids are recomputed using the least square optimization technique.

Centroid initialization methods

Positioning the initial centroids can be challenging and the aim is to initialize centroids as close as possible to optimal values of actual centroids. It is recommended to use some strategies for defining initial centroids as it directly impacts the overall runtime. The traditional way is to select the centroids randomly but there are other methods as well which we will cover in the section.

- Random Data Points

  → This is the traditional approach of initializing centroids where K random data points are selected and defined as centroids.

→ As we saw in the above example, in this method each data instance in the dataset will have to be enumerated and will have to keep a record of the minimum/maximum value of each attribute.

→ This is a time–consuming process; with increased dataset complexity the number of steps to achieve the correct centroid or correct cluster will also increase.

- Naive Sharding

→ The sharding centroid initialization algorithm primarily depends on the composite summation value of all the attributes for a particular instance or row in a dataset.

→ The idea is to calculate the composite value and then use it to sort the instances of the data. Once the data set is sorted, it is then divided horizontally into k shards.

→ Finally, all the attributes from each shard will be summed and their mean will be calculated.

→ The shard attributes mean value collection will be identified as the set of centroids that can be used for initialization.

→ Centroid initialization using sharding happens in linear time and the resultant execution time is much better than random centroid initialization.

Applications of K-Means Clustering

K–Means clustering is used in a variety of examples or business cases in real life, like:

→ Academic performance

→ Diagnostic systems

→ Search engines

→ Wireless sensor networks

Types of Clustering

Clustering is a type of unsupervised learning wherein data points are grouped into different sets based on their degree of similarity.

The various types of clustering are:

→ Hierarchical clustering

→ Partitioning clustering

Hierarchical clustering is further subdivided into:

→ Agglomerative clustering

→ Divisive clustering

Partitioning clustering is further subdivided into:

→ K–Means clustering

Distance Measure

Distance measure determines the similarity between two elements and influences the shape of clusters.

K−Means clustering supports various kinds of distance measures, such as:

→ Euclidean distance measure

→ Manhattan distance measure

→ A squared euclidean distance measure

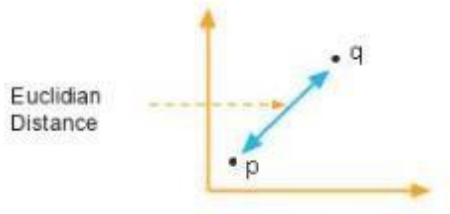
→ Cosine distance measure

Euclidean Distance Measure

The most common case is determining the distance between two points. If we have a point P and point Q, the euclidean distance is an ordinary straight line. It is the distance between the two points in Euclidean space.

The formula for distance between two points is shown below:

$$d=\sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}$$



Squared Euclidean Distance Measure

This is identical to the Euclidean distance measurement but does not take the square root at the end. The formula is shown below:

$$d=\sum_{i=1}^{n}(q_i-p_i)^2$$

Manhattan Distance Measure

The Manhattan distance is the simple sum of the horizontal and vertical components or the distance between two points measured along axes at right angles. Note that we are taking the absolute value so that the negative values don't come into play.

The formula is shown below:

$$d=\sum_{i=1}^{n}|q_x-p_x|+|q_x-p_y|$$