**Artificial Intelligence and Machine Learning – AIML / AI&ML – (CS3491)**

**PART-C**

**1. Explain the Mini-Max algorithm and how it is work for game tic-tac-toe. (dec03,04, May -09,10, 17 , 19)**

 Mini-max algorithm is a recursive or backtracking algorithm which is used in decisionmaking and game theory. It provides an optimal move for the player assuming that opponent is

also playing optimally.

 Mini-Max algorithm uses recursion to search through the game-tree.

 Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tactoe, go, and various tow-players game. This Algorithm computes the minimax decision for the

current state.

 In this algorithm two players play the game, one is called MAX and other is called MIN.

 Both the players fight it as the opponent player gets the minimum benefit while they get the

maximum benefit.

 Both Players of the game are opponent of each other, where MAX will select the maximized

value and MIN will select the minimized value.

 The minimax algorithm performs a depth-first search algorithm for the exploration of the

complete game tree.

 The minimax algorithm proceeds all the way down to the terminal node of the tree, then

backtrack the tree as the recursion.

Pseudo-code for MinMax Algorithm:

function minimax(node, depth, maximizingPlayer) is

if depth ==0 or node is a terminal node then

return static evaluation of node

if MaximizingPlayer then // for Maximizer Player

maxEva= -infinity

for each child of node do

eva= minimax(child, depth-1, false)

maxEva= max(maxEva,eva) //gives Maximum of the values

return maxEva

else // for Minimizer player

minEva= +infinity

for each child of node do

eva= minimax(child, depth-1, true)

minEva= min(minEva, eva) //gives minimum of the values

return minEva

Working of Min-Max Algorithm:

o The working of the minimax algorithm can be easily described using an example. Below we

have taken an example of game-tree which is representing the two-player game.

o In this example, there are two players one is called Maximizer and other is called Minimizer.

o Maximizer will try to get the Maximum possible score, and Minimizer will try to get the

minimum possible score.

o This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves

to reach the terminal nodes.

o At the terminal node, the terminal values are given so we will compare those value and

backtrack the tree until the initial state occurs. Following are the main steps involved in

solving the two-player game tree:

Step-1: In the first step, the algorithm generates the entire game-tree and applies the utility function

to get the utility values for the terminal states. In the below tree diagram, let's take A is the initial state

of the tree. Suppose maximizer takes first turn which has worst-case initial value =- infinity, and

minimizer will take next turn which has worst-case initial value = +infinity.

Step 2: Now, first we find the utilities value for the Maximizer, its initial value is -∞, so we will

compare each value in terminal state with initial value of Maximizer and determines the higher nodes

values. It will find the maximum among the all.

o For node D max(-1,- -∞) => max(-1,4)= 4

o For Node E max(2, -∞) => max(2, 6)= 6

o For Node F max(-3, -∞) => max(-3,-5) = -3

o For node G max(0, -∞) = max(0, 7) = 7

Step 3: In the next step, it's a turn for minimizer, so it will compare all nodes value with +∞, and will

find the 3rd layer node values.

o For node B= min(4,6) = 4

o For node C= min (-3, 7) = -3

Step 4: Now it's a turn for Maximizer, and it will again choose the maximum of all nodes value and

find the maximum value for the root node. In this game tree, there are only 4 layers, hence we reach

immediately to the root node, but in real games, there will be more than 4 layers.

o For node A max(4, -3)= 4

Properties of Mini-Max

algorithm:

o Complete- Min-Max algorithm is

Complete. It will definitely find a

solution (if exist), in the finite

search tree.

o Optimal- Min-Max algorithm is

optimal if both opponents are

playing optimally.

o Time complexity- As it performs

DFS for the game-tree, so the

time complexity of Min-Max

algorithm is O(bm

), where b is branching factor of the game-tree, and m is the maximum

depth of the tree.

o Space Complexity- Space complexity of Mini-max algorithm is also similar to DFS which

is O(bm).

Limitation of the minimax Algorithm:

The main drawback of the minimax algorithm is that it gets really slow for complex games such as

Chess, go, etc. This type of games has a huge branching factor, and the player has lots of choices to

decide. This limitation of the minimax algorithm can be improved from alpha-beta pruning.

Des cribing Minimax

The key to the Minimax algorithm is a back and forth between the two players, where

the player whose "turn it is" desires to pick the move with the maximum score. In turn,

the scores for each of the available moves are determined by the opposing player

deciding which of its available moves has the minimum score. And the scores for the

opposing players moves are again determined by the turn-taking player trying to

maximize its score and so on all the way down the move tree to an end state.

A description for the algorithm, assuming X is the "turn taking player," would look

something like:

 If the game is over, return the score from X's perspective.

 Otherwise get a list of new game states for every possible move

 Create a scores list

 For each of these states add the minimax result of that state to the scores list

 If it's X's turn, return the maximum score from the scores list

 If it's O's turn, return the minimum score from the scores list

You'll notice that this algorithm is recursive; it flips back and forth between the

players until a final score is found. Let’s walk through the algorithm's execution with

the full move tree, and show why, algorithmically, the instant winning move will be

picked:

 It's X's turn in state 1. X generates the states 2, 3, and 4 and calls minimax on

those states.

 State 2 pushes the score of +10 to state 1's score list, because the game is in an

end state.

 State 3 and 4 are not in end states, so 3 generates states 5 and 6 and calls

minimax on them, while state 4 generates states 7 and 8 and calls minimax on

them.

 State 5 pushes a score of -10 onto state 3's score list, while the same happens

for state 7 which pushes a score of -10 onto state 4's score list.

 State 6 and 8 generate the only available moves, which are end states, and so

both of them add the score of +10 to the move lists of states 3 and 4.

 Because it is O's turn in both state 3 and 4, O will seek to find the minimum

score, and given the choice between -10 and +10, both states 3 and 4 will yield

-10.

 Finally the score list for states 2, 3, and 4 are populated with +10, -10 and -10

respectively, and state 1 seeking to maximize the score will chose the winning

move with score +10, state 2.

That is certainly a lot to take in. And that is why we have a computer execute this

algorithm.

**2. Explain heuristic search with an example. Explain A\* search and give**

**the proof of optimality of A\*.**

**Informed Search Algorithms**

 Informed search algorithm contains an array of knowledge such as how far we are from the

goal, path cost, how to reach to goal node, etc. This knowledge help agents to explore less to the

search space and find more efficiently the goal node.

 The informed search algorithm is more useful for large search space. Informed search algorithm

uses the idea of heuristic, so it is also called Heuristic search.

Heuristics function: 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. The heuristic method, however, might not always give the best solution,

but it guaranteed to find a good solution in reasonable time. Heuristic function estimates how close a

state is to the goal. It is represented by h(n), and it calculates the cost of an optimal path between the

pair of states. The value of the heuristic function is always positive.

Admissibility of the heuristic function is given as:

h (n) <= h\*(n)

Here h (n) is heuristic cost, and h\*(n) is the estimated cost. Hence heuristic cost should be

less than or equal to the estimated cost.

Pure Heuristic Search:

Pure heuristic search is the simplest form of heuristic search algorithms. It expands nodes based

on their heuristic value h(n). It maintains two lists, OPEN and CLOSED list. In the CLOSED list, it

places those nodes which have already expanded and in the OPEN list, it places nodes which have yet

not been expanded. On each iteration, each node n with the lowest heuristic value is expanded and

generates all its successors and n is placed to the closed list. The algorithm continues unit a goal state

is found.

In the informed search we will discuss two main algorithms which are given below:

o Best First Search Algorithm(Greedy search)

o A\* Search Algorithm

 Best-first Search Algorithm (Greedy Search):

Greedy best-first search algorithm always selects the path which appears best at that moment. It

is the combination of depth-first search and breadth-first search algorithms. It uses the heuristic

function and search. Best-first search allows us to take the advantages of both algorithms. With the

help of best-first search, at each step, we can choose the most promising node. In the best first search

algorithm, we expand the node which is closest to the goal node and the closest cost is estimated by

heuristic function, i.e.

f (n)= g(n).

Where, h(n)= estimated cost from node n to the goal.

The greedy best first algorithm is implemented by the priority queue.

Best first search algorithm:

o Step 1: Place the starting node into the OPEN list.

o Step 2: If the OPEN list is empty, Stop and return failure.

o Step 3: Remove the node n, from the OPEN list which has the lowest value of h(n), and

places it in the CLOSED list.

o Step 4: Expand the node n, and generate the successors of node n.

o Step 5: Check each successor of node n, and find whether any node is a goal node or not. If

any successor node is goal node, then return success and terminate the search, else proceed to

Step 6.

o Step 6: For each successor node, algorithm checks for evaluation function f(n), and then

check if the node has been in either OPEN or CLOSED list. If the node has not been in both

list, then add it to the OPEN list.

o Step 7: Return to Step 2.

Advantages:

o Best first search can switch between BFS and DFS by gaining the advantages of both the

algorithms.

o This algorithm is more efficient than BFS and DFS algorithms.

Disadvantages:

o It can behave as an unguided depth-first search in the worst case scenario.

o It can get stuck in a loop as DFS.

o This algorithm is not optimal.

Example:o Consider the below search problem, and we will traverse it using greedy best-first search. At

each iteration, each node is expanded using evaluation function f(n)=h(n) , which is given in

the below table.

In this search example, we are using two lists which are OPEN and CLOSED Lists. Following are

the iteration for traversing the above example.

Expand the nodes of S and put in the CLOSED list

Initialization: Open [A, B], Closed [S]

Iteration 1: Open [A], Closed [S, B]Iteration2: Open[E,F,A],Closed[S,B]

: Open [E, A], Closed [S, B, F]

Iteration 3: Open [I, G, E, A], Closed [S, B, F]

: Open [I, E, A], Closed [S, B, F, G]

Hence the final solution path will be: S----> B----->F----> G

Time Space Complexity: The worst case space complexity of Greedy

best first search is O(bm). Where, m is the maximum depth of the

search space.

Complete: Greedy best-first search is also incomplete, even if the given state space is finite.

Optimal: Greedy best first search algorithm is not optimal.

2.) A\* Search Algorithm:

A\* search is the most commonly known form of best-first search. It uses heuristic function h(n),

and cost to reach the node n from the start state g(n). It has combined features of UCS and greedy

best-first search, by which it solve the problem efficiently. A\* search algorithm finds the shortest path

through the search space using the heuristic function. This search algorithm expands less search tree

and provides optimal result faster. A\* algorithm is similar to UCS except that it uses g(n)+h(n)

instead of g(n). In A\* search algorithm, we use search heuristic as well as the cost to reach the node.

Hence we can combine both costs as following, and this sum is called as a fitness number.

Complexity: The worst case time complexity of Greedy best first search is O(bm

).

Algorithm of A\* search:

Step1: Place the starting node in the OPEN list.

Step 2: Check if the OPEN list is empty or not, if the list is empty then return failure and stops.

Step 3: Select the node from the OPEN list which has the smallest value of evaluation function (g+h),

if node n is goal node then return success and stop, otherwise

Step 4: Expand node n and generate all of its successors, and put n into the closed list. For each

successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute

evaluation function for n' and place into Open list

Step 5: Else if node n' is already in OPEN and CLOSED, then it should be attached to the back

pointer which reflects the lowest g(n') value.

Step 6: Return to Step 2.

Advantages:

o A\* search algorithm is the best algorithm than other search algorithms.

o A\* search algorithm is optimal and complete.

o This algorithm can solve very complex problems.

Disadvantages:

o It does not always produce the shortest path as it mostly based on heuristics and

approximation.

o A\* search algorithm has some complexity issues.

o The main drawback of A\* is memory requirement as it keeps all generated nodes in the

memory, so it is not practical for various large-scale problems.

Points to remember:

o A\* algorithm returns the path which occurred first, and it does not search for all remaining

paths.

o The efficiency of A\* algorithm depends on the quality of heuristic.

o A\* algorithm expands all nodes which satisfy the condition f(n)<="" li="">

Complete: A\* algorithm is complete as long as:

o Branching factor is finite.

o Cost at every action is fixed.

Optimal: A\* search algorithm is optimal if it follows below two conditions:

o Admissible: the first condition requires for optimality is that h(n) should be an admissible

heuristic for A\* tree search. An admissible heuristic is optimistic in nature.

o Consistency: Second required condition is consistency for only A\* graph-search.

If the heuristic function is admissible, then A\* tree search will always find the least cost path.

Time Complexity: The time complexity of A\* search algorithm depends on heuristic function, and

the number of nodes expanded is exponential to the depth of solution d. So the time complexity is

O(b^d), where b is the branching factor.

Space Complexity: The space complexity of A\* search algorithm is O(b^d)

**UNIT – 2**

**1. How to handle uncertain knowledge with example? And How to represent**

**knowledge in an uncertain domain? (Dec- 2013) and Define uncertain knowledge,**

**prior probability and conditional probability. State the Baye’s theorem. How it is**

**useful for decision making under uncertainty? (May- 2014)**

**PROBABILISTIC REASONING**

Uncertainty :

Till now, we have learned knowledge representation using first-order logic and propositional logic

with certainty, which means we were sure about the predicates. With this knowledge representation,

we might write A→B, which means if A is true then B is true, but consider a situation where we are

not sure about whether A is true or not then we cannot express this statement, this situation is called

uncertainty. So to represent uncertain knowledge, where we are not sure about the predicates, we need

uncertain reasoning or probabilistic reasoning.

Causes of uncertainty:

Following are some leading causes of uncertainty to occur in the real world.

1. Information occurred from unreliable sources.

2. Experimental Errors

3. Equipment fault

4. Temperature variation

5. Climate change.Probabilistic reasoning:

Probabilistic reasoning is a way of knowledge representation where we apply the concept of

probability to indicate the uncertainty in knowledge. In probabilistic reasoning, we combine

probability theory with logic to handle the uncertainty.

 We use probability in probabilistic reasoning because it provides a way to handle the

uncertainty that is the result of someone's laziness and ignorance.

 In the real world, there are lots of scenarios, where the certainty of something is not

confirmed, such as "It will rain today," "behavior of someone for some situations," "A match

between two teams or two players."

 These are probable sentences for which we can assume that it will happen but not sure about

it, so here we use probabilistic reasoning.

Need of probabilistic reasoning in AI:

o When there are unpredictable outcomes.

o When specifications or possibilities of predicates becomes too large to handle.

o When an unknown error occurs during an experiment.

In probabilistic reasoning, there are two ways to solve problems with uncertain knowledge:

o Bayes' rule

o Bayesian Statistics

As probabilistic reasoning uses probability and related terms, so before understanding probabilistic

reasoning, let's understand some common terms:

Probability: Probability can be defined as a chance that an uncertain event will occur. It is the

numerical measure of the likelihood that an event will occur. The value of probability always remains

between 0 and 1 that represent ideal uncertainties.

0 ≤ P(A) ≤ 1, where P(A) is the probability of an event A.

P(A) = 0, indicates total uncertainty in an event A.

P(A) =1, indicates total certainty in an event A.

We can find the probability of an uncertain event by using the below formula.

o P(¬A) = probability of a not happening event.

o P(¬A) + P(A) = 1.

Event: Each possible outcome of a variable is called an event.

Sample space: The collection of all possible events is called sample space.

Random variables: Random variables are used to represent the events and objects in the real world.

Prior probability: The prior probability of an event is probability computed before observing new

information.

Posterior Probability: The probability that is calculated after all evidence or information has taken

into account. It is a combination of prior probability and new information.

Conditional probability:

Conditional probability is a probability of occurring an event when another event has already

happened.

Let's suppose, we want to calculate the event A when event B has already occurred, "the

probability of A under the conditions of B", it can be written as:

Where P (A/ B) = Joint probability of a and B

P (B) = Marginal probability of B.

If the probability of A is given and we need to find the probability of B, then it will be given as:

It can be explained by using the below Venn diagram, where B is occurred event, so sample space

will be reduced to set B, and now we can only calculate event A when event B is already occurred by

dividing the probability of P(A⋀B) by P( B ).

Bayes' theorem in Artificial intelligence

Bayes' theorem:

Bayes' theorem is also known as Bayes' rule, Bayes' law, or Bayesian reasoning, which determines

the probability of an event with uncertain knowledge. In probability theory, it relates the conditional

probability and marginal probabilities of two random events. Baye’s theorem was named after the

British mathematician Thomas Bayes. The Bayesian inference is an application of Baye’s theorem,

which is fundamental to Bayesian statistics. It is a way to calculate the value of P(B|A) with the

knowledge of P(A|B).

 Bayes' theorem allows updating the probability prediction of an event by observing new

information of the real world.

 Example: If cancer corresponds to one's age then by using Bayes' theorem, we can determine the

probability of cancer more accurately with the help of age.

 Bayes' theorem can be derived using product rule and conditional probability of event A with

known event B: As from product rule we can write:The above equation (a) is called as Bayes' rule or Bayes' theorem. This equation is basic of most

modern AI systems for probabilistic inference.

It shows the simple relationship between joint and conditional probabilities. Here,

 P (A|B) is known as posterior, which we need to calculate, and it will be read as Probability

of hypothesis A when we have occurred an evidence B.

 P (B|A) is called the likelihood, in which we consider that hypothesis is true, then we

calculate the probability of evidence.

 P(A) is called the prior probability, probability of hypothesis before considering the

evidence

 P (B) is called marginal probability, pure probability of evidence.

In the equation (a), in general, we can write P (B) = P(A)\*P(B|Ai), hence the Bayes' rule can be

written as:

Where A1, A2, A3,........, An is a set of mutually exclusive and exhaustive events.

Applying Bayes' rule:

Bayes' rule allows us to compute the single term P(B|A) in terms of P(A|B), P(B), and P(A). This is

very useful in cases where we have a good probability of these three terms and want to determine the

fourth one. Suppose we want to perceive the effect of some unknown cause, and want to compute that

cause, then the Bayes' rule becomes:

Application of Bayes' theorem:

o It is used to calculate the next step of the robot when the already executed step is given.

o Bayes' theorem is helpful in weather forecasting.

o It can solve the Monty Hall problem.

**UNIT – 3**

**1. Explain the types of learning in machine learning.**

**Types of Machine Learning**

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

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:

o Classification

o 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. 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:

o Random Forest Algorithm

o Decision Tree Algorithm

o Logistic Regression Algorithm

o Support Vector Machine Algorithm

b) 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:

o Simple Linear Regression Algorithm

o Multivariate Regression Algorithm

o Decision Tree Algorithm

o Lasso Regression

Advantages:

o Since supervised learning work with the labelled dataset so we can have an exact idea about

the classes of objects.

o These algorithms are helpful in predicting the output on the basis of prior experience.

Disadvantages:

o These algorithms are not able to solve complex tasks.

o It may predict the wrong output if the test data is different from the training data.

o It requires lots of computational time to train the algorithm.

Applications of Supervised Learning

Some common applications of Supervised Learning are given below:

oImageSegmentation

oMedicalDiagnosis

oFraud Detection

oSpam detection Speech Recognition

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:

o Clustering

o Association

Advantages:

o These algorithms can be used for complicated tasks compared to the supervised ones because

these algorithms work on the unlabeled dataset.

o Unsupervised algorithms are preferable for various tasks as getting the unlabeled dataset is

easier as compared to the labelled dataset.

Disadvantages:

o 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.

o 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

**2. Explain in details about regression models- linear classification**

**models.**

**Logistic Regression in Machine Learning**

o Logistic regression is one of the most popular Machine Learning algorithms, which comes

under the Supervised Learning technique. It is used for predicting the categorical dependent

variable using a given set of independent variables.

o Logistic regression predicts the output of a categorical dependent variable. Therefore the

outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or

False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values

which lie between 0 and 1.

o Logistic Regression is much similar to the Linear Regression except that how they are used.

Linear Regression is used for solving Regression problems, whereas Logistic regression is

used for solving the classification problems.

o In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic

function, which predicts two maximum values (0 or 1).o The curve from the logistic function indicates the likelihood of something such as whether the

cells are cancerous or not, a mouse is obese or not based on its weight, etc.

o Logistic Regression is a significant machine learning algorithm because it has the ability to

provide probabilities and classify new data using continuous and discrete datasets.

o Logistic Regression can be used to classify the observations using different types of data and

can easily determine the most effective variables used for the classification. The below image

is showing the logistic function:

Logistic Function (Sigmoid Function):

o The sigmoid function is a mathematical function used to map the predicted values to

probabilities.

o It maps any real value into another value within a range of 0 and 1.

o The value of the logistic regression must be between 0 and 1, which cannot go beyond this

limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or

the logistic function.

o In logistic regression, we use the concept of the threshold value, which defines the probability

of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the

threshold values tends to 0.

Assumptions for Logistic Regression:

o The dependent variable must be categorical in nature.

o The independent variable should not have multi-collinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The

mathematical steps to get Logistic Regression equations are given below:

o We know the equation of the straight line can be written as:

o In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above

equation by (1-y):

o But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will

become:

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression:

Logistic Regression can be classified into three types:

o 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.

o Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered

types of the dependent variable, such as "cat", "dogs", or "sheep"

o Ordinal: In ordinal Logistic regression, there can be 3 or more possible ordered types of

dependent variables, such as "low", "Medium", or "High".

**UNIT – 4**

**1. Explain details about KNN algorithm?**

**K-Nearest Neighbor(KNN) Algorithm for Machine Learning**

o K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on

Supervised Learning technique.

o K-NN algorithm assumes the similarity between the new case/data and available cases and put

the new case into the category that is most similar to the available categories.

o K-NN algorithm stores all the available data and classifies a new data point based on the

similarity. This means when new data appears then it can be easily classified into a well suite

category by using K- NN algorithm.

o K-NN algorithm can be used for Regression as well as for Classification but mostly it is used

for the Classification problems.

o K-NN is a non-parametric algorithm, which means it does not make any assumption on

underlying data.

o It is also called a lazy learner algorithm because it does not learn from the training set

immediately instead it stores the dataset and at the time of classification, it performs an action

on the dataset.

o KNN algorithm at the training phase just stores the dataset and when it gets new data, then it

classifies that data into a category that is much similar to the new data.

o Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we

want to know either it is a cat or dog. So for this identification, we can use the KNN

algorithm, as it works on a similarity measure. Our KNN model will find the similar features

of the new data set to the cats and dogs images and based on the most similar features it will

put it in either cat or dog category

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data

point x1, so this data point will lie in which of these categories. To solve this type of problem, we

need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a

particular dataset. Consider the below diagram:

How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

o Step-1: Select the number K of the neighbors

o Step-2: Calculate the Euclidean distance of K number of neighbors

o Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

o Step-4: Among these k neighbors, count the number of the data points in each category.

o Step-5: Assign the new data points to that category for which the number of the neighbor is

maximum.

o Step-6: Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below

image:

o Firstly, we will choose the number of neighbors, so we will choose the k=5.

o Next, we will calculate the Euclidean distance between the data points. The Euclidean

distance is the distance between two points, which we have already studied in geometry. It can

be calculated as:

o By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors

in category A and two nearest neighbors in category B. Consider the below image:o As we can see the 3 nearest neighbors are from category A, hence this new data point must

belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

o There is no particular way to determine the best value for "K", so we need to try some values

to find the best out of them. The most preferred value for K is 5.

o A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in

the model.

o Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

o It is simple to implement.

o It is robust to the noisy training data

o It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

o Always needs to determine the value of K which may be complex some time.

o The computation cost is high because of calculating the distance between the data points for

all the training samples.

**2. Explain in detail about Gaussian mixture models and**

**expectation maximization?**

**EM algorithm in GMM**

In statistics, EM (expectation maximization) algorithm handles latent variables, while GMM is the

Gaussian mixture model.

 Gaussian mixture models (GMMs) are a type of machine learning algorithm. They are used to

classify data into different categories based on the probability distribution. Gaussian mixture

models can be used in many different areas, including finance, marketing and so much more.

 Gaussian Mixture Models (GMMs) give us more flexibility than K-Means. With GMMs we

assume that the data points are Gaussian distributed; this is a less restrictive assumption than

saying they are circular by using the mean. That way, we have two parameters to describe the

shape of the clusters: the mean and the standard deviation!

 Taking an example in two dimensions, this means that the clusters can take any kind of

elliptical shape (since we have standard deviation in both the x and y directions). Thus, each

Gaussian distribution is assigned to a single cluster. In order to find the parameters of the

Gaussian for each cluster (e.g the mean and standard deviation) we will use an optimization

algorithm called Expectation–Maximization (EM). Take a look at the graphic below as an

illustration of the Gaussians being fitted to the clusters. Then we can proceed on to the process

of Expectation–Maximization clustering using GMMs.

 Gaussian mixture models (GMM) are a probabilistic concept used to model real-world data

sets. GMMs are a generalization of Gaussian distributions and can be used to represent any

data set that can be clustered into multiple Gaussian distributions. The Gaussian mixture

model is a probabilistic model that assumes all the data points are generated from a mix of

Gaussian distributions with unknown parameters.

 A Gaussian mixture model can be used for clustering, which is the task of grouping a set of

data points into clusters. GMMs can be used to find clusters in data sets where the clusters

may not be clearly defined. Additionally, GMMs can be used to estimate the probability that a

new data point belongs to each cluster. Gaussian mixture models are also relatively robust to

outliers, meaning that they can still yield accurate results even if there are some data points

that do not fit neatly into any of the clusters. This makes GMMs a flexible and powerful tool

for clustering data.

 It can be understood as a probabilistic model where Gaussian distributions are assumed for

each group and they have means and co variances which define their parameters. GMM

consists of two parts – mean vectors (μ) & covariance matrices (Σ). A Gaussian distribution is

defined as a continuous probability distribution that takes on a bell-shaped curve. Another name for Gaussian distribution is the normal distribution. Here is a picture of Gaussian

mixture models:

 GMM has many applications, such as density estimation, clustering, and image segmentation.

For density estimation, GMM can be used to estimate the probability density function of a set

of data points. For clustering, GMM can be used to group together data points that come from

the same Gaussian distribution. And for image segmentation, GMM can be used to partition

an image into different regions.

 Gaussian mixture models can be used for a variety of use cases, including identifying

customer segments, detecting fraudulent activity, and clustering images. In each of these

examples, the Gaussian mixture model is able to identify clusters in the data that may not be

immediately obvious. As a result, Gaussian mixture models are a powerful tool for data

analysis and should be considered for any clustering task.

Expectation-maximization (EM) method in relation to GMM

In Gaussian mixture models, an expectation-maximization method is a powerful tool for

estimating the parameters of a Gaussian mixture model (GMM). The expectation is termed E and

maximization is termed M. Expectation is used to find the Gaussian parameters which are used to

represent each component of gaussian mixture models. Maximization is termed M and it is involved

in determining whether new data points can be added or not.

 The expectation-maximization method is a two-step iterative algorithm that alternates

between performing an expectation step, in which we compute expectations for each data

point using current parameter estimates and then maximize these to produce a new gaussian,

followed by a maximization step where we update our gaussian means based on the maximum

likelihood estimate.

 The EM method works by first initializing the parameters of the GMM, then iteratively

improving these estimates. At each iteration, the expectation step calculates the expectation of

the log-likelihood function with respect to the current parameters. This expectation is then

used to maximize the likelihood in the maximization step. The process is then repeated until

convergence. Here is a picture representing the two-step iterative aspect of the algorithm

The EM algorithm consists of two steps: the E- step and the M-step. Firstly, the model parameters and

the can be randomly initialized. In the E-step, the algorithm tries to guess the value of based on the

parameters, while in the M-step, the algorithm updates the value of the model parameters based on

the guess of the E-step. These two steps are repeated until convergence is reached. The algorithm in

GMM is repeat until convergence.

Optimization uses the Expectation Maximization algorithm, which alternates between two

steps:

1. E-step: Compute the posterior probability over z given our current model - i.e. how much

do we think each Gaussian generates each datapoint.

2. M-step: Assuming that the data really was generated this way, change the parameters of

each Gaussian to maximize the probability that it would generate the data it is currently

responsible for.

The K-Means Algorithm:

1.Assignment step: Assign each data point to the closest cluster

2. Refitting step: Move each cluster center to the center of gravity of the data assigned to it

The EM Algorithm:

1. E-step: Compute the posterior probability over z given our current model

2. M-step: Maximize the probability that it would generate the data it is currently responsible for.

**UNIT – 5**

**1.Explain about Error Backpropagation**?

Backpropagation, or backward propagation of errors, is an algorithm that is designed to test for

errors working back from output nodes to input nodes. It is an important mathematical tool for

improving the accuracy of predictions in data mining and machine learning. Essentially,

backpropagation is an algorithm used to calculate derivatives quickly.There are two leading types of backpropagation networks:

1. Static backpropagation:

Static backpropagation is a network developed to map static inputs for static outputs. Static

backpropagation networks can solve static classification problems, such as optical character

recognition (OCR).

2. Recurrent backpropagation.

The recurrent backpropagation network is used for fixed-point learning. Recurrent

backpropagation activation feeds forward until it reaches a fixed value.

What is a backpropagation algorithm in a neural network?

Artificial neural networks use backpropagation as a learning algorithm to compute a gradient descent

with respect to weight values for the various inputs. By comparing desired outputs to achieved system

outputs, the systems are tuned by adjusting connection weights to narrow the difference between the

two as much as possible. The algorithm gets its name because the weights are updated backward,

from output to input.

Advantages

 It does not have any parameters to tune except for the number of inputs.

 It is highly adaptable and efficient and does not require any prior knowledge about the

network.

 It is a standard process that usually works well.

 It is user-friendly, fast and easy to program.

 Users do not need to learn any special functions.

DISADVANTAGES

It prefers a matrix-based approach over a mini-batch approach.

 Data mining is sensitive to noise and irregularities.

 Performance is highly dependent on input data.

 Training is time- and resource-intensive.

Features of Backpropagation:

1. it is the gradient descent method as used in the case of simple perceptron network with the

differentiable unit.

2. it is different from other networks in respect to the process by which the weights are

calculated during the learning period of the network.

3. training is done in the three stages :

 the feed-forward of input training pattern

 the calculation and backpropagation of the error

 updation of the weight

Working of Backpropagation:

Neural networks use supervised learning to generate output vectors from input vectors that the

network operates on. It Compares generated output to the desired output and generates an error

report if the result does not match the generated output vector. Then it adjusts the weights

according to the bug report to get your desired output.

Backpropagation Algorithm:

Step 1: Inputs X, arrive through the preconnected path.

Step 2: The input is modeled using true weights W. Weights are usually chosen randomly.

Step 3: Calculate the output of each neuron from the input layer to the hidden layer to the output

layer.

Step 4: Calculate the error in the outputs

Backpropagation Error= Actual Output – Desired Output

Step 5: From the output layer, go back to the hidden layer to adjust the weights to reduce the error.

Step 6: Repeat the process until the desired output is achieved.

Fig: Error backpropagation

 x = inputs training vector x=(x1,x2,…………xn).

 t = target vector t=(t1,t2……………tn).

 δk = error at output unit.

 δj = error at hidden layer.

 α = learning rate.

 V0j = bias of hidden unit j.

Training Algorithm :

Step 1: Initialize weight to small random values.Step 2: While the steps stopping condition is to be false do step 3 to 10.

Step 3: For each training pair do step 4 to 9 (Feed-Forward).

Step 4: Each input unit receives the signal unit and transmits the signal xi signal to all the units.

Step 5: Each hidden unit Zj (z=1 to a) sums its weighted input signal to calculate its net input

zinj = v0j + Σxivij ( i=1 to n)

Applying activation function zj = f(zinj) and sends this signals to all units in the layer about

i.e output units

For each output l=unit yk = (k=1 to m) sums its weighted input signals.

yink = w0k + Σ ziwjk (j=1 to a)

and applies its activation function to calculate the output signals.

yk = f(yink)

Backpropagation Error :

Step 6: Each output unit yk (k=1 to n) receives a target pattern corresponding to an input pattern

then error is calculated as:

δk = ( tk – yk ) + yink

Step 7: Each hidden unit Zj (j=1 to a) sums its input from all units in the layer above

δinj = Σ δj wjk

The error information term is calculated as :

δj = δinj + zinj

Updation of weight and bias :

Step 8: Each output unit yk (k=1 to m) updates its bias and weight (j=1 to a). The weight correction

term is given by :

Δ wjk = α δk zj

and the bias correction term is given by Δwk = α δk.

therefore wjk(new) = wjk(old) + Δ wjk

w0k(new) = wok(old) + Δ wok

for each hidden unit zj (j=1 to a) update its bias and weights (i=0 to n) the weight

connection term

Δ vij = α δj xi

and the bias connection on term

Δ v0j = α δj

Therefore vij(new) = vij(old) + Δvij

v0j(new) = v0j(old) + Δv0j

Step 9: Test the stopping condition. The stopping condition can be the minimization of error,

number of epochs.