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

**PART-B**

**UNIT1**

**1.Discuss any 2 uninformed search methods with examples. (Dec-2009), explain the following uninformed search strategies. I) IDDFS AND 2) Bidirectional search. (May 2010); what is uninformed search and explain depth first search with example. (May-2013, Dec 2013 , may 2014; May 2015, Dec 2016**)

An Uninformed search is a group of wide range usage algorithms of the era. These algorithms are brute force operations, and they don’t have extra information about the search space; the only information they have is on how to traverse or visit the nodes in the tree. Thus uninformed search algorithms are also called blind search algorithms. The search algorithm produces the search tree without using any domain knowledge, which is the brute force in nature. They are different from informed search algorithms in a way that you check for a goal when a node is generated or expanded, and they don’t have any background information on how to approach the goal.

**TYPES OF UNINFORMED SEARCH ALGORITHMS**

 **Breadth-First Search Algorithms**

BFS is a search operation for finding the nodes in a tree. The algorithm works breadth wise and

traverses to find the desired node in a tree. It starts searching operation from the root nodes and

expands the successor nodes at that level before moving ahead and then moves along breadth wise for

further expansion.

 It occupies a lot of memory space, and time to execute when the solution is at the bottom or

end of the tree and uses the FIFO queue.

 Time Complexity of BFS is expressed as T (n) = 1+n2+n3+…….+ nd= O (nd

) and;

 Space Complexity of BFS is O (nd

).

 The breadth-first search algorithm is complete.

 The optimal solution is possible to obtain from BFS.

**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

**Depth Limited Search Algorithm**

The DLS algorithm is one of the uninformed strategies. A depth limited search is close to DFS to

some extent. It can find the solution to the demerit of DFS. The nodes at the depth may behave as if

no successor exists at the depth. Depth-limited search can be halted in two cases:

o SFV: The Standard failure value which tells that there is no solution to the problem.

o CFV: The Cutoff failure value tells that there is no solution within the given depth.

 The DLS is efficient in memory space utilization.

 Time Complexity is expressed as O(bℓ

).

 Space Complexity is expressed as O(b×ℓ).

 It has the demerit of incompleteness. It is complete only if the solution is above the depth

limit.

**Uniform-cost Search Algorithm**

The UCS algorithm is used for visiting the weighted tree. The main goal of the uniform cost search is

to fetch a goal node and find the true path, including the cumulative cost. The following are the

properties of the UCS algorithm:

 The expansion takes place on the basis of cost from the root. The UCS is implemented using a

priority queue.

 The UCS does not care for the number of steps, and so it may end up an infinite loop.

 The uniform-cost search algorithm is known to be complete.

 Time Complexity can be expressed as O(b1 + [C\*/ε])/

 Space Complexity is expressed as O(b1 + [C\*/ε]).

 We can say that UCs is the optimal algorithm as it chooses the path with the lowest cost only.

**Iterative deepening depth-first Search**

This algorithm is a combination of BFS and DFS searching techniques. It is iterative in nature. The

best depth is found using it. The algorithm is set to search only at a certain depth. The depth keeps

increasing at each recursive step until it finds the goal node.

 The power of BFS and DFS combination is observed in this algorithm.

 When the search space is large, it proves itself, and the depth is not known.

 This algorithm has one demerit, and it is that it iterates all the previous steps.

 The algorithm is known to be complete only if the branching factor is known r finite.

 Time Complexity is expressed as O(bd

).

 Space Complexity is expressed as O(bd).

 This algorithm is optimal.

**Bidirectional Search Algorithm**

The Two way or Bidirectional search algorithm executes in a way that t has to run two searches

simultaneously one in a forward direction and the other in the backward direction. The search will

stop when the two simultaneous searches intersect each other to find the goal node. It is free to use

any search algorithm discussed above, like BFS, DFS, etc.

 Bidirectional search is quick and occupies less memory.

 The implementation is difficult, and the goal node should be known in advance to execute it.

 The Bidirectional Search algorithm is found to be complete and optimal.

 Time Complexity is expressed as O(bd

).

 Space Complexity is expressed as O(bd).

**2. Explain Alpha-Beta Pruning using example. (Dec—04,10, May-10,17)**

**Alpha-Beta Pruning**

o Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization

technique for the minimax algorithm.

o 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.

o 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.

o The two-parameter can be defined as:

a. 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 -∞.

b. 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 +∞.

o 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: α>=β

Key points about alpha-beta pruning:

o The Max player will only update the value of alpha.

o The Min player will only update the value of beta.

o While backtracking the tree, the node values will be passed to upper nodes instead of values

of alpha and beta.

o 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 α= -∞ and β= +∞,

these value of alpha and beta passed down to node B where again α= -∞ and β= +∞, 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 algorithms backtrack to node B, where the value of β will change as this is a turn of

Min, Now β= +∞ will compare with the available subsequent nodes value, i.e. min (∞, 3) = 3, hence

at node B now α= -∞, and β= 3. In the next step, algorithm traverse the next successor of Node B

which is node E, and the values of α= -∞, and β= 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.

**Move Ordering in Alpha-Beta pruning:**

The effectiveness of alpha-beta pruning is highly dependent on the order in which each node is

examined. Move order is an important aspect of alpha-beta pruning. It can be of two types:

o Worst ordering: In some cases, alpha-beta pruning algorithm does not prune any of the

leaves of the tree, and works exactly as minimax algorithm. In this case, it also consumes

more time because of alpha-beta factors, such a move of pruning is called worst ordering. In

this case, the best move occurs on the right side of the tree. The time complexity for such an

order is O(bm

).

o Ideal ordering: The ideal ordering for alpha-beta pruning occurs when lots of pruning

happens in the tree, and best moves occur at the left side of the tree. We apply DFS hence it

first search left of the tree and go deep twice as minimax algorithm in the same amount of

time. Complexity in ideal ordering is O (bm/2).

**UNIT – 2**

**1. Discuss about Bayesian theory and Bayesian network. (Dec 2017)**

**Bayesian network**

o "A Bayesian network is a probabilistic graphical model which represents a set of variables and

their conditional dependencies using a directed acyclic graph."

o It is also called a Bayes network, belief network, decision network, or Bayesian model.

o 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:

o Directed Acyclic Graph

o 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:

o Each node corresponds to the random variables, and a variable can be continuous or discrete.

o 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

o In the above diagram, A, B, C, and D are random variables represented by the nodes of

the network graph.

o 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.

o Node C is independent of node A.

**The Bayesian network has mainly two components:**

o Causal Component

o 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 (Xi|Xi-1,........., X1) = P(Xi |Parents(Xi ))

**2. Explain about Dempster shafer theory. (May- 2017)**

Dempster – Shafer Theory (DST)

o DST is a mathematical theory of evidence based on belief functions and plausible reasoning.

It is used to combine separate pieces of information (evidence) to calculate the probability of

an event.

o DST offers an alternative to traditional probabilistic theory for the mathematical

representation of uncertainty.

o DST can be regarded as, a more general approach to represent uncertainty than the Bayesian

approach. Bayesian methods are sometimes inappropriate

Example:

Let A represent the proposition "Moore is attractive". Then the axioms of probability insist

that P(A) + P(¬A) = 1. Now suppose that Andrew does not even know who "Moore" is, then Also, it

is not fair to say that he disbelieves the proposition. It would therefore be meaningful to denote

Andrew's belief B of B(A) and B(¬A) as both being 0.

Dempster-Shafer Model

The idea is to allocate a number between 0 and 1 to indicate a degree of belief on a proposal as

in the probability framework. However, it is not considered a probability but a belief mass. The

distribution of masses is called basic belief assignment.

In other words, in this formalism a degree of belief (referred as mass) is represented as

a belief function rather than a Bayesian probability distribution.

Example: Belief assignment

Suppose a system has five members, say five independent states, and exactly one of which is actual.

If the original set is called S, | S | = 5, then the set of all subsets (the power set) is called 2

S

. If each

possible subset as a binary vector (describing any member is present or not by writing 1 or 0 ),

then 2

5

subsets are possible, ranging from the empty subset ( 0, 0, 0, 0, 0 ) to the "everything" subset (

1, 1, 1, 1, 1 ).

The "empty" subset represents a "contradiction", which is not true in any state, and is thus assigned a

mass of one; The remaining masses are normalized so that their total is 1. The "everything" subset is

labeled as "unknown"; it represents the state where all elements are present one , in the sense that you

cannot tell which is actual.

Belief and Plausibility

Shafer's framework allows for belief about propositions to be represented as intervals, bounded by

two values, belief (or support) and plausibility:

belief ≤ plausibility

Belief in a hypothesis is constituted by the sum of the masses of all sets enclosed by it (i.e. the sum of

the masses of all subsets of the hypothesis). It is the amount of belief that directly supports a given

hypothesis at least in part, forming a lower bound.

Plausibility is 1 minus the sum of the masses of all sets whose intersection with the hypothesis is

empty. It is an upper bound on the possibility that the hypothesis could possibly happen, up to that

value, because there is only so much evidence that contradicts that hypothesis.

Example:

A proposition say "the cat in the box is dead." Suppose we have belief of 0.5 and plausibility of

0.8 for the proposition.

For example,

Suppose we have a belief of 0.5 for a proposition, say "the cat in the box is dead." This means that

we have evidence that allows us to state strongly that the proposition is true with a confidence of 0.5.

However, the evidence contrary to that hypothesis (i.e. "the cat is alive") only has a confidence of 0.2.

The remaining mass of 0.3 (the gap between the 0.5 supporting evidence on the one hand, and the 0.2

contrary evidence on the other) is "indeterminate," meaning that the cat could either be dead or alive.

This interval represents the level of uncertainty based on the evidence in the system.

The "neither" hypothesis is set to zero by definition (it corresponds to "no solution"). The

orthogonal hypotheses "Alive" and "Dead" have probabilities of 0.2 and 0.5, respectively.

This could correspond to "Live/Dead Cat Detector" signals, which have respective reliabilities

of 0.2 and 0.5.

Finally, the all-encompassing "Either" hypothesis (which simply acknowledges there is a cat in

the box) picks up the slack so that the sum of the masses is 1. The belief for the "Alive" and

"Dead" hypotheses matches their corresponding masses because they have no subsets; belief

for "Either" consists of the sum of all three masses (Either, Alive, and Dead) because "Alive"

and "Dead" are each subsets of "Either".

The "Alive" plausibility is 1 − m (Dead): 0.5 and the "Dead" plausibility is 1 − m (Alive): 0.8.

In other way, the "Alive" plausibility is m(Alive) + m(Either) and the "Dead" plausibility

is m(Dead) + m(Either).

Finally, the "Either" plausibility sums m(Alive) + m(Dead) + m(Either). The universal

hypothesis ("Either") will always have 100% belief and plausibility—it acts as a checksum of

sorts.

Plausibility in K: It is the sum of masses of set that intersects with K.

i.e; Pl(K) = m(a) + m(b) + m(c) + m(a, b) + m(b, c) + m(a, c) + m(a, b, c)

Characteristics of Dempster Shafer Theory:

 It will ignorance part such that probability of all events aggregate to 1.

 Ignorance is reduced in this theory by adding more and more evidences.

 Combination rule is used to combine various types of possibilities.

Advantages:

 As we add more information, uncertainty interval reduces.

 DST has much lower level of ignorance.

 Diagnose hierarchies can be represented using this.

 Person dealing with such problems is free to think about evidences.

Disadvantages:

 In this, computation effort is high, as we have to deal with 2n

of sets.

**UNIT – 3**

1. **Explain in details about regression models- linear regression models?**

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:

y= a0+a1x+ ε

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:

o 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.

o MultipleLinearregression :

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 Xaxis, 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 Xaxis, 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 functiono 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:

Gradient Descent:

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

o A regression model uses gradient descent to update the coefficients of the line by reducing the

cost function.

o 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.

**2. Explain in detail about support vector machine?**

**Support Vector Machine Algorithm**

 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.

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

**Types of SVM**

**SVM can be of two types:**

**o 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.

**o 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.

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:

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=x2

+y2

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. Since we are in 3-d Space, hence it is looking like a plane

parallel to the x-axis. If we convert it in 2d space with z=1.

**UNIT – 4**

**1.Explain the various ensemble learning techniques?**

Ensemble methods are techniques that aim at improving the accuracy of results in models by

combining multiple models instead of using a single model. The combined models increase the

accuracy of the results significantly. This has boosted the popularity of ensemble methods in machine

learning.

Categories of Ensemble Methods

Ensemble methods fall into two broad categories, i.e., sequential ensemble techniques and

parallel ensemble techniques. Sequential ensemble techniques generate base learners in a sequence,

e.g., Adaptive Boosting (AdaBoost). The sequential generation of base learners promotes the

dependence between the base learners. The performance of the model is then improved by assigning

higher weights to previously misrepresented learners.

 In parallel ensemble techniques, base learners are generated in a parallel format,

e.g., random forest. Parallel methods utilize the parallel generation of base learners to

encourage independence between the base learners. The independence of base learners

significantly reduces the error due to the application of averages.

 The majority of ensemble techniques apply a single algorithm in base learning, which results

in homogeneity in all base learners. Homogenous base learners refer to base learners of the

same type, with similar qualities. Other methods apply heterogeneous base learners, giving

rise to heterogeneous ensembles. Heterogeneous base learners are learners of distinct types.

Main Types of Ensemble Methods

1. Bagging

 Bagging, the short form for bootstrap aggregating, is mainly applied in classification

and regression. It increases the accuracy of models through decision trees, which reduces

variance to a large extent. The reduction of variance increases accuracy, eliminating

overfitting, which is a challenge to many predictive models.

 Bagging is classified into two types, i.e., bootstrapping and aggregation. Bootstrapping is a

sampling technique where samples are derived from the whole population (set) using the

replacement procedure. The sampling with replacement method helps make the selection

procedure randomized. The base learning algorithm is run on the samples to complete the

procedure.

 Aggregation in bagging is done to incorporate all possible outcomes of the prediction and

randomize the outcome. Without aggregation, predictions will not be accurate because all

outcomes are not put into consideration. Therefore, the aggregation is based on the probability

bootstrapping procedures or on the basis of all outcomes of the predictive models.

Bagging is advantageous since weak base learners are combined to form a single strong learner

that is more stable than single learners. It also eliminates any variance, thereby reducing the overfitting of models. One limitation of bagging is that it is computationally expensive. Thus, it can

lead to more bias in models when the proper procedure of bagging is ignored.

**2. Boosting**

 Boosting is an ensemble technique that learns from previous predictor mistakes to make better

predictions in the future. The technique combines several weak base learners to form one

strong learner, thus significantly improving the predictability of models. Boosting works by

arranging weak learners in a sequence, such that weak learners learn from the next learner in

the sequence to create better predictive models.

 Boosting takes many forms, including gradient boosting, Adaptive Boosting (AdaBoost), and

XGBoost (Extreme Gradient Boosting). AdaBoost uses weak learners in the form of decision

trees, which mostly include one split that is popularly known as decision stumps. AdaBoost’s

main decision stump comprises observations carrying similar weights.

 Gradient boosting adds predictors sequentially to the ensemble, where preceding predictors

correct their successors, thereby increasing the model’s accuracy. New predictors are fit tocounter the effects of errors in the previous predictors. The gradient of descent helps the

gradient booster identify problems in learners’ predictions and counter them accordingly.

**3. Stacking**

Stacking, another ensemble method is often referred to as stacked generalization. This technique

works by allowing a training algorithm to ensemble several other similar learning algorithm

predictions. Stacking has been successfully implemented in regression, density estimations, distance

learning, and classifications. It can also be used to measure the error rate involved during bagging.

Variance Reduction

Ensemble methods are ideal for reducing the variance in models, thereby increasing the accuracy

of predictions. The variance is eliminated when multiple models are combined to form a single

prediction that is chosen from all other possible predictions from the combined models. An ensemble

of models combines various models to ensure that the resulting prediction is the best possible, based

on the consideration of all predictions.

Simple Ensemble Techniques

In this section, we will look at a few simple but powerful techniques, namely:

1. Max Voting

2. Averaging

3. Weighted Averaging

**2. Explain in detail about k-means algorithm?**

**K-Means Clustering Algorithm**

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering

problems in machine learning or data science. In this topic, we will learn what is K-means clustering

algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

What is K-Means Algorithm?

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset

into different clusters. Here K defines the number of pre-defined clusters that need to be created in the

process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

 It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a

way that each dataset belongs only one group that has similar properties.

 It allows us to cluster the data into different groups and a convenient way to discover the

categories of groups in the unlabeled dataset on its own without the need for any training. It is a centroid-based algorithm, where each cluster is associated with a centroid. The main

aim of this algorithm is to minimize the sum of distances between the data point and their

corresponding clusters.

 The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of

clusters, and repeats the process until it does not find the best clusters. The value of k should

be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

o Determines the best value for K center points or centroids by an iterative process.

o Assigns each data point to its closest k-center. Those data points which are near to the

particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters. The

below diagram explains the working of the K-means Clustering Algorithm:

How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which mean reassign each datapoint to the new closest centroid of

each cluster.Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

Let's understand the above steps by considering the visual plots:

o Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is

given below: Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them

into different clusters. It means here we will try to group these datasets into two different

clusters.

o We need to choose some random k points or centroid to form the cluster. These points can be

either the points from the dataset or any other point. So, here we are selecting the below two

points as k points, which are not the part of our dataset. Consider the below image:

Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will

compute it by applying some mathematics that we have studied to calculate the distance between two

points. So, we will draw a median between both the centroids. Consider the below image:

From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and

points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for

clear visualization. As we need to find the closest cluster, so we will repeat the process by choosing a

new centroid. To choose the new centroids, we will compute the center of gravity of these centroids,

and will find new centroids as below: Next, we will reassign each datapoint to the e new centroid. For

this, we will repeat the same process of finding a median line.

How to choose the value of "K number of clusters" in K-means Clustering?

The performance of the K-means clustering algorithm depends upon highly efficient clusters that

it forms. But choosing the optimal number of clusters is a big task. There are some different ways to

find the optimal number of clusters, but here we are discussing the most appropriate method to find

the number of clusters or value of K. The method is given below:

**UNIT – 5**

**1.Explain in detail about Perceptrons and its types?**

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.

Basic Components of Perceptron

Mr. Frank Rosenblatt invented the perceptron model as a binary classifier which contains three

o 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.

o Weight 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.

o 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:

o Sign function

o Step function, and

o Sigmoid function

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.

Multi-Layered 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:

o Forward Stage: Activation functions start from the input layer in the forward stage and terminate

on the output layer.

o 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.

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.

Advantages of Multi-Layer Perceptron:

o A multi-layered perceptron model can be used to solve complex non-linear problems.

o It works well with both small and large input data.

o It helps us to obtain quick predictions after the training.

o It helps to obtain the same accuracy ratio with large as well as small data.

Disadvantages of Multi-Layer Perceptron:

o In Multi-layer perceptron, computations are difficult and time-consuming.

o In multi-layer Perceptron, it is difficult to predict how much the dependent variable affects

each independent variable.

o The model functioning depends on the quality of the training.

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.

Characteristics of Perceptron

The perceptron model has the following characteristics.

1. Perceptron is a machine learning algorithm for supervised learning of binary classifiers.

2. In Perceptron, the weight coefficient is automatically learned.

3. Initially, weights are multiplied with input features, and the decision is made whether the

neuron is fired or not.

4. The activation function applies a step rule to check whether the weight function is greater than

zero.

5. The linear decision boundary is drawn, enabling the distinction between the two linearly

separable classes +1 and -1.

6. If the added sum of all input values is more than the threshold value, it must have an output

signal; otherwise, no output will be shown.

**2.Explain detail about activation functions?**

**ACTIVATION FUNCTIONS**

The activation function decides whether a neuron should be activated or not by

calculating the weighted sum and further adding bias to it. The purpose of the activation function

is to introduce non-linearity into the output of a neuron.

 In a neural network, we would update the weights and biases of the neurons on the basis of

the error at the output. This process is known as back-propagation. Activation functions make

the back-propagation possible since the gradients are supplied along with the error to update

the weights and biases.

 A neural network without an activation function is essentially just a linear regression

model. The activation function does the non-linear transformation to the input making it

capable to learn and perform more complex tasks.

Calculation at Output layer

z(2) = (W(2) \* [W(1)X + b(1)]) + b(2)

z(2) = [W(2) \* W(1)] \* X + [W(2)\*b(1) + b(2)]

Let,

[W(2) \* W(1)] = W

[W(2)\*b(1) + b(2)] = b

Final output : z(2) = W\*X + b

which is again a linear function

Variants of Activation Function

Linear Function

 Equation : Linear function has the equation similar to as of a straight line i.e. y = x

 No matter how many layers we have, if all are linear in nature, the final activation function

of last layer is nothing but just a linear function of the input of first layer.

 Range : -inf to +inf

 Uses: Linear activation function is used at just one place i.e. output layer.

 Issues: If we will differentiate linear function to bring non-linearity, result will no more

depend on input “x” and function will become constant, it won’t introduce any groundbreaking behavior to our algorithm.

For example: Calculation of price of a house is a regression problem. House price may have any

big/small value, so we can apply linear activation at output layer. Even in this case neural net must

have any non-linear function at hidden layers.

Sigmoid Function

 It is a function which is plotted as ‘S’ shaped graph.

 Equation : A = 1/(1 + e-x)

 Nature: Non-linear. Notice that X values lies between -2 to 2, Y values are very steep. This

means, small changes in x would also bring about large changes in the value of Y.

 Value Range : 0 to 1

 Uses: Usually used in output layer of a binary classification, where result is either 0 or 1,

as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to

be 1 if value is greater than 0.5 and 0 otherwise.

Tanh Function

 The activation that works almost always better than sigmoid function is Tanh function also

knows as Tangent Hyperbolic function. It’s actually mathematically shifted version of the

sigmoid function. Both are similar and can be derived from each other.

 Equation :-

 Value Range :- -1 to +1

 Nature :- non-linear

 Uses: - Usually used in hidden layers of a neural network as its values lies between -1 to

1 hence the mean for the hidden layer comes out be 0 or very close to it, hence helps

in centering the data by bringing mean close to 0. This makes learning for the next layer much

easier.RELU Function

 It Stands for Rectified linear unit. It is the most widely used activation function. Chiefly

implemented in hidden layers of neural network.

 Equation: - A(x) = max (0, x). It gives an output x if x is positive and 0 otherwise.

 Value Range :- [0, inf)

 Nature: - non-linear, which means we can easily backpropagate the errors and have

multiple layers of neurons being activated by the ReLU function.

 Uses: - ReLu is less computationally expensive than tanh and sigmoid because it involves

simpler mathematical operations. At a time only a few neurons are activated making the

network sparse making it efficient and easy for computation.

In simple words, RELU learns much faster than sigmoid and Tanh function.

Softmax Function

The softmax function is also a type of sigmoid function but is handy when we are trying to

handle multi- class classification problems.

 Nature :- non-linear

 Uses: - Usually used when trying to handle multiple classes. The softmax function was

commonly found in the output layer of image classification problems. The softmax function

would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of

the outputs.

 Output: - The softmax function is ideally used in the output layer of the classifier where

we are actually trying to attain the probabilities to define the class of each input.

 The basic rule of thumb is if you really don’t know what activation function to use, then

simply use RELU as it is a general activation function in hidden layers and is used in most

cases these days.

 If your output is for binary classification then, sigmoid function is very natural choice for

output layer.

 If your output is for multi-class classification then, Softmax is very useful to predict the

probabilities of each classes.

**2.Explain in detail about gradient descent optimization?**

**Gradient descent optimization**

 Gradient descent is an optimization algorithm in gadget mastering used to limit a feature with the

aid of iteratively moving towards the minimal fee of characteristic.

 We essentially use this algorithm when we have to locate the least possible values which could

fulfill a given free function. In gadget getting to know, greater regularly that not we try to limit loss

features (like mean squared error). By minimizing the loss characteristic, we will improve our model

and gradient descent is one of the most popular algorithms used for this cause.

 The graph above shows how exactly a gradient descent set of rules works. We first take a factor in the value function and begin shifting in steps in the direction of the

minimum factor. The size of the step, or how quickly we ought to converge to the minimum factor is

defined by learning rate.

 We can cowl more location with better learning fee but at the risk of overshooting the minima. On

the opposite hand, small steps/ smaller gaining knowledge of charges will eat a number of times to

attain the lowest point.

 Now, the direction where in algorithm has to transport is also important. We calculate this by way

of using derivatives. You need to be familiar with derivatives from calculus. A spinoff is largely

calculated because the slope of the graph at any specific factor. We get that with the aid of finding the

tangent line to the graph at that point. The extra sleep the tangent, would suggest that more steps

would be needed to reach minimum point; much less steep might suggest lesser steps are required to

reach the minimum factor.

Fig: Gradient descent optimization

Stochastic gradient descent

The word stochastic means a system or a process that is linked with a random probability. Hence, in

stochastic gradient descent, a few samples are selected randomly instead of the whole data set for

each iteration.

 Stochastic gradient descent is a type of gradient descent that runs one training example per

iteration. It processes a training epoch for each example within a dataset and updates each

training example’s parameters one at a time.

 As it requires only one training example at a time, hence it is easier to store in allocated

memory. However, it shows some computational efficiency losses in comparison to batch

gradient systems as it shows frequent updates that require more detail and speed. Further, due to frequent updates, it is also treated as a noisy gradient. However, sometimes it

can be helpful in finding the global minimum and also escaping the local minimum.

Advantages of stochastic gradient descent:

It is easier to allocate in desired memory.

It is relatively fast to compute than batch gradient descent.

It is more efficient for large dataset.

Disadvantages of stochastic gradient descent:

SGD require a number of hyperparameters such as the regularization parameter and the

number of iterations.

SGD is sensitive to feature scaling.