

GEETANJALI INSTITUTE OF TECHNICAL STUDIES

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



MACHINE LEARNING

Model Papers(1-5) With Solution

MODEL PAPER - 1

PART-A

Q1.What are the Different Types of Machine Learning?

Ans. Machine Learning is a subset of artificial intelligence which focuses mainly on machine learning from their experience and making predictions based on its experience.

Types of Machine Learning:-

- (1) Supervised Learning
- (2) Unsupervised Learning
- (3) Reinforcement Learning

Q2.What is Overfitting and how can you avoid it?

Ans. Overfitting :- A statistical model is said to be overfitted, when we train it with a lot of data. When a model gets trained with so much of data, it starts learning from the noise and inaccurate data entries in our data set. Then the model does not categorize the data correctly, because of too much of details and noise.

How to avoid Overfitting: The commonly used methodologies are:

- (1). Cross- Validation
- (2). Early Stopping
- (3). Pruning
- (4). Regularization

Q3.How do you handle missing or corrupted data in a dataset?

Ans. (1). Deleting Rows

- (2) Replacing With Mean/Median/Mode
- (3) Assigning An Unique Category
- (4) Predicting The Missing Values
- (5) Using Algorithms Which Support Missing Values

Q4. How can you choose a classifier based on a training set data size?

Ans. In general, it depends on the kind of data and amount of samples x features. For instance, I would recommend to use naive Bayes or linear SVM for text classification/ categorization. For datasets with numerical attributes: I would suggest linear SVM, neural networks or logistic regression if the amount of features is much greater than the number of samples. On the other hand, I would recommend neural networks or SVM with RBF or polynomial kernel if the amount of samples is not too large and greater than the number of features. Otherwise, if the number of samples is huge I would suggest to use neural networks or linear SVM, and so on.

Q5.Explain the Confusion Matrix with respect to machine learning algorithms.

Ans. Confusion Matrix: A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

Q6. What are the three stages of building a model in machine learning?

Ans. The three stages to build the hypotheses in machine learning are model building, model testing and applying model.

Q7.What is Deep Learning?

Ans. Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

Q8.What are the differences between Machine Learning and Deep Learning?

	Machine Learning	Deep Learning
How does it work?	Uses types of automated algorithms which learn to predict future decisions and model functions using the data fed to it.	Interprets data features and its relationships using neural networks which pass the relevant information through several stages of data processing.
Management	The various algorithms are directed by the analysts to examine the different variables in the datasets.	Once they are implemented, the algorithms are usually self-directed for the relevant data analysis.
Number of Data Points	Usually, there are a few thousand data points used for the analysis.	There are a few million data points used for the analysis.
Output	The output is usually a numerical value, like a score or a classification.	The output can be anything from a score, an element, free text or sound, etc.

Q9. What are the applications of Supervised Machine Learning in modern businesses?

Ans.Marketing and Sales

Lifetime Value

Churn

Sentiment analysis

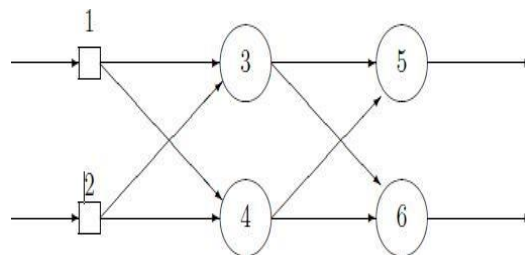
Recommendations

Q10.What is Semi-supervised Machine Learning?

Ans. Semi-supervised Learning:- This kind of learning is used and applied to the same kind of scenarios where supervised learning is applicable. However, one must note that this technique uses both unlabeled and labeled data for training. Ideally, a small set of labeled data, along with a large volume of unlabeled data is used, as it takes less time, money and efforts to acquire unlabeled data. This type of machine learning is often used with methods, such as regression, classification and prediction. Companies that usually find it challenging to meet the high costs associated with labeled training process opt for semi-supervised learning.

PART-B

Q1. The following diagram represents a feed-forward neural network with one hidden layer:



A weight on connection between nodes i and j is denoted by w_{ij} , such as w_{13} is the weight on the connection between nodes 1 and 3. The following table lists all the weights in the network:

$w_{13} = -2$	$w_{35} = 1$
$w_{23} = 3$	$w_{45} = -1$
$w_{14} = 4$	$w_{36} = -1$
$w_{24} = -1$	$w_{46} = 1$

Each of the nodes 3, 4, 5 and 6 uses the following activation function: $\Phi(v) =$

$$\left\{ \begin{array}{l} 1 \text{ if } v \geq 0 \\ \text{Otherwise} \end{array} \right.$$

where v denotes the weighted sum of a node. Each of the input nodes (1 and 2) can only receive binary values (either 0 or 1). Calculate the output of the network (y_5 and y_6) for each of the input

Pattern:	P_1	P_2	P_3	P_4
Node 1:	0	1	0	1
Node 2:	0	0	1	1

patterns:

Ans. In order to find the output of network it is necessary to calculate weighted sums of hidden nodes 3 and 4:

$$v_3 = w_{13}x_1 + w_{23}x_2, \quad v_4 = w_{14}x_1 + w_{24}x_2$$

Then find the outputs from hidden nodes using activation function ϕ :

$$y_3 = \phi(v_3), \quad y_4 = \phi(v_4).$$

Use the outputs of the hidden nodes y_3 and y_4 as the input values to the output layer (nodes 5 and 6), and find weighted sums of output nodes 5 and 6:

$$v_5 = w_{35}y_3 + w_{45}y_4, \quad v_6 = w_{36}y_3 + w_{46}y_4.$$

Finally, find the outputs from nodes 5 and 6 (also using ϕ):

$$y_5 = \phi(v_5), \quad y_6 = \phi(v_6).$$

The output Pattern will be (y_5, y_6). Perform these calculation for each input pattern:

P_1 : Input pattern (0, 0)

$$v_3 = -2 \cdot 0 + 3 \cdot 0 = 0, \quad y_3 = \phi(0) = 1$$

$$v_4 = 4 \cdot 0 - 1 \cdot 0 = 0, \quad y_4 = \phi(0) = 1$$

$$v_5 = 1 \cdot 1 - 1 \cdot 1 = 0, \quad y_5 = \phi(0) = 1$$

$$v_6 = -1 \cdot 1 + 1 \cdot 1 = 0, \quad y_6 = \phi(0) = 1$$

The output of the network is (1, 1).

P_2 : Input pattern (1, 0)

$$v_3 = -2 \cdot 1 + 3 \cdot 0 = -2, \quad y_3 = \phi(-2) = 0$$

$$v_4 = 4 \cdot 1 - 1 \cdot 0 = 4, \quad y_4 = \phi(4) = 1$$

$$v_5 = 1 \cdot 0 - 1 \cdot 1 = -1, \quad y_5 = \phi(-1) = 0$$

$$v_6 = -1 \cdot 0 + 1 \cdot 1 = 1, \quad y_6 = \phi(1) = 1$$

The output of the network is (0, 1).

P_3 : Input pattern (0, 1)

$$v_3 = -2 \cdot 0 + 3 \cdot 1 = 3, \quad y_3 = \phi(3) = 1$$

$$v_4 = 4 \cdot 0 - 1 \cdot 1 = -1, \quad y_4 = \phi(-1) = 0$$

$$v_5 = 1 \cdot 1 - 1 \cdot 0 = 1, \quad y_5 = \phi(1) = 1$$

$$v_6 = -1 \cdot 1 + 1 \cdot 0 = -1, \quad y_6 = \phi(-1) = 0$$

The output of the network is (1, 0).

P4: Input pattern (1, 1)

$$v_3 = -2 \cdot 1 + 3 \cdot 1 = 1, \quad y_3 = \phi(1) = 1$$

$$v_4 = 4 \cdot 1 - 1 \cdot 1 = 3, \quad y_4 = \phi(3) = 1$$

$$v_5 = 1 \cdot 1 - 1 \cdot 1 = 0, \quad y_5 = \phi(0) = 1$$

$$v_6 = -1 \cdot 1 + 1 \cdot 1 = 0, \quad y_6 = \phi(0) = 1$$

The output of the network is (1, 1).

Q2.(a) What is a training set and how is it used to train neural networks?

Ans. Training set is a set of pairs of input patterns with corresponding desired output patterns. Each pair represents how the network is supposed to respond to a particular input. The network is trained to respond correctly to each input pattern from the training set. Training algorithms that use training sets are called supervised learning algorithms. We may think of a supervised learning as learning with a teacher, and the training set as a set of examples. During training the network, when presented with input patterns, gives 'wrong' answers (not desired output). The error is used to adjust the weights in the network so that next time the error was smaller. This procedure is repeated using many examples (pairs of inputs and desired outputs) from the training set until the error becomes sufficiently small.

Q2.(b) Describe the main steps of the supervised training algorithm?

Ans. Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.^[1] It infers a function from *labeled training data* consisting of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances

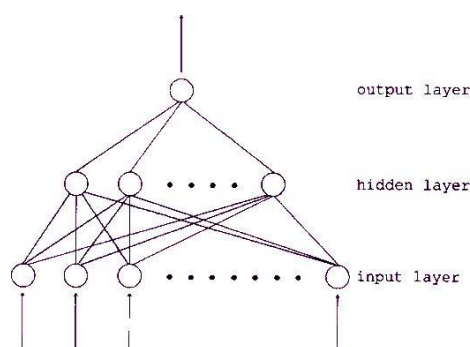
In order to solve a given problem of supervised learning, one has to perform the following steps:

1. Determine the type of training examples. Before doing anything else, the user should decide what kind of data is to be used as a training set. In the case of handwriting analysis, for example, this might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.
2. Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.

3. Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.
4. Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support vector machines or decision trees.
5. Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a *validation* set) of the training set, or via cross-validation.
6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set. Training data already trained.

Q3. Consider a multilayer feed forward neural network. Enumerate and explain steps in back propagation algorithm use to train network.

Ans. Multilayer feed Forward Neural Network :- MLF neural networks, trained with a back-propagation learning algorithm, are the most popular neural networks. They are applied to a wide variety of chemistry related problems.



A MLF neural network consists of neurons, that are ordered into layers. The first layer is called the input layer, the last layer is called the output layer, and the layers between are hidden layers. For the formal description of the neurons we can use the so-called mapping function I , that assigns for each neuron i a subset $N(i) \subset V$ which consists of all ancestors of the given neuron.

$$x_i = f(\xi_i) \quad (1)$$

$$\xi_i = \vartheta_i + \sum_{j \in \Gamma_i^{-1}} \omega_{ij} x_j \quad (2)$$

where ξ_i is the potential of the i th neuron and function $f(\cdot)$ is the so-called transfer function (the summation in Eq. (2) is carried out over all neurons j transferring the signal to the i th neuron). The threshold coefficient can be understood as a weight coefficient of the connection with formally added neuron y , where x_I — (so-called bias).

Back-propagation training algorithm:-

```

initialize network weights (often small random values)
do
  forEach training example named ex

```

```

prediction = neural-net-output(network, ex) // forward pass
actual = teacher-output(ex)
compute error (prediction - actual) at the output units
compute {displaystyle \Delta w_{h}} for all weights from hidden layer to output layer // backward pass
compute {displaystyle \Delta w_{i}} for all weights from input layer to hidden layer // backward pass
continued
update network weights // input layer not modified by error estimate
until all examples classified correctly or another stopping criterion satisfied
return the network

```

Q4.Explain Naïve bayes classifier with example.

Ans. A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

Bayes Theorem:

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

Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

Example:-

	OUTLOOK	TEMPRATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

We classify whether the day is suitable for playing golf, given the features of the day. The columns represent these features and the rows represent individual entries. If we take the first row of the dataset, we can observe that is not suitable for playing golf if the outlook is rainy, temperature is hot, humidity is high and it is not windy. We make two assumptions here, one as stated above we consider that these predictors are independent. That is, if the temperature is hot, it does not necessarily mean that the humidity is high. Another assumption made here is that all the predictors have an equal effect on the outcome. That is, the day being windy does not have more importance in deciding to play golf or not.

According to this example, Bayes theorem can be rewritten as:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

\mathbf{X} is given as,

$$X = (x_1, x_2, x_3, \dots, x_n)$$

Here x_1, x_2, \dots, x_n represent the features, i.e they can be mapped to outlook, temperature, humidity and windy. By substituting for \mathbf{X} and expanding using the chain rule we get,

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y)\dots P(x_n|y)P(y)}{P(x_1)P(x_2)\dots P(x_n)}$$

Outlook

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%

Q5. Enumerate the steps in K-nearest neighbour algorithm.

Ans. The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

The KNN Algorithm

1. Load the data
2. Initialize K to your chosen number of neighbors
3. 3. For each example in the data

Calculate the distance between the query example and the current example from the data.

Add the distance and the index of the example to an ordered collection

4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection.
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

Pros and Cons of KNN

Pros

- It is very simple algorithm to understand and interpret.
- It is very useful for nonlinear data because there is no assumption about data in this algorithm.
- It is a versatile algorithm as we can use it for classification as well as regression.
- It has relatively high accuracy but there are much better supervised learning models than KNN.

Cons

- It is computationally a bit expensive algorithm because it stores all the training data.
- High memory storage required as compared to other supervised learning algorithms.
- Prediction is slow in case of big N.
- It is very sensitive to the scale of data as well as irrelevant features.

Q6.Explain perceptron and Delta training rule.

Ans. Delta rule: When the neuron is trained via the delta rule, the algorithm is:

1. Evaluate the network according to the equation: $\Sigma = x_1 w_1 + x_2 w_2 - \theta$.
2. If the current output y is already equal to the desired output t , repeat step 1 with a different set of inputs. Otherwise, proceed to step 4.
3. Adjust the current weights according to $\Delta w_i = \alpha(t^p - y^p)x_i^p$, where Δw_i is the change in the neuron's weight, α is the learning rate, t^p is the target output given the input set p , y^p is the real output of the neuron given the input set p after being passed through the threshold set by the bias θ , and x_i is the i^{th} input.
4. Repeat the algorithm from step 1 until $y = t$ for every vector pair.

Perceptron training rule: When the perceptron training rule algorithm is selected, the steps are:

1. Evaluate the network according to the equation: $\Sigma = x_1 \omega_1 + x_2 \omega_2 - \theta$.
2. If the result of step 1 is greater than zero, $y = 1$; if it is less than zero, $y = 0$.
3. If the current output y is already equal to the desired output t , repeat step 1 with a different set of inputs. If the current output y is different from the desired output t , proceed to step 4.
4. Adjust the current weights according to: $\Delta \omega_i = \alpha(t^p - a^p)x_i^p$, where $\Delta \omega_i$ is the change in the neuron's weight, α is the learning rate, t^p is the target output given the input set p , a^p is the real output of the neuron given the input set p without being passed through the threshold set by the bias θ , and x_i is the i^{th} input.
5. Repeat the algorithm from step 1 until $y = t$ for every vector pair.

Q7. Describe these terms in brief.

(1)PCA hypothesis:- The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent. The same is done by transforming the variables to a new set of variables, which are known as the principal components (or simply, the PCs) and are orthogonal, ordered such that the retention of variation present in the original variables decreases as we move down in the order. So, in this way, the 1st principal component retains maximum variation that was present in the original components. The principal components are the eigenvectors of a covariance matrix, and hence they are orthogonal.

Dimensionality (get sample code): It is the number of random variables in a dataset or simply the number of features, or rather more simply, the number of columns present in your dataset.

Correlation (get sample code): It shows how strongly two variable are related to each other. The value of the same ranges for -1 to +1. Positive indicates that when one variable increases, the other increases as well, while negative indicates the other decreases on increasing the former. And the modulus value of indicates the strength of relation.

Orthogonal: (get sample code) Uncorrelated to each other, i.e., correlation between any pair of variables is 0.

Eigenvectors: (get sample code) Eigenvectors and Eigenvalues are in itself a big domain, let's restrict ourselves to the knowledge of the same which we would require here. So, consider a non-zero vector v . It is an eigenvector of a square matrix A , if $A v$ is a scalar multiple of v . Or simply: $A v = \lambda v$

Here, v is the eigenvector and λ is the eigenvalue associated with it.

Covariance Matrix: This matrix consists of the covariances between the pairs of variables. The (i,j) th element is the covariance between i -th and j -th variable.

Properties of Principal Component

1. The PCs are essentially the linear combinations of the original variables, the weights vector in this combination is actually the eigenvector found which in turn satisfies the principle of least squares.

2. The PCs are orthogonal, as already discussed.

3. The variation present in the PCs decrease as we move from the 1st PC to the last one, hence the importance.

(2.)Mistake bound model of learning:- In the MB model, learning is in stages. In each stage:

1. The learner gets an unlabeled example.
2. The learner predicts its classification.
3. The learner is told the correct label.

DEFINITION: Algorithm A has mistake-bound M for learning class C if A makes at most M mistakes on any sequence that is consistent with a function in C. [later we'll talk about extending to when there's a "mostly consistent" f in C]

Notice that since we're not assuming anything about how examples are selected, we can't necessarily talk about "how much data do I need to converge". E.g., maybe we end up seeing the same example over and over again and don't learn anything new. But, that's OK because we (hopefully) won't be making mistakes either. Later on, when we talk about distributional models, we will be able to convert mistake bounds into bounds on how much data we need to converge. But the

mistake-bound setting is nice because it's very clean -- no probabilities. So this is the toughest model for positive results.

We'll think of A as being good if its mistake bound is polynomial in n = the size of the examples, and s = the description length of the smallest consistent function (for classes like DNF and Decision Trees

that can have very complicated functions in them). [We are thinking of a "concept class" as both a set of functions and a description language for them.]

DEFINITION: C is *learnable* in the mistake bound model if there exists an algorithm with mistake bound $\text{poly}(n,s)$, and furthermore whose running time per stage is $\text{poly}(n,s)$.

PART-C

Q1. What is the procedure of building Decision Tree using ID3 algorithm with Gain and Entropy. Illustrate with example.

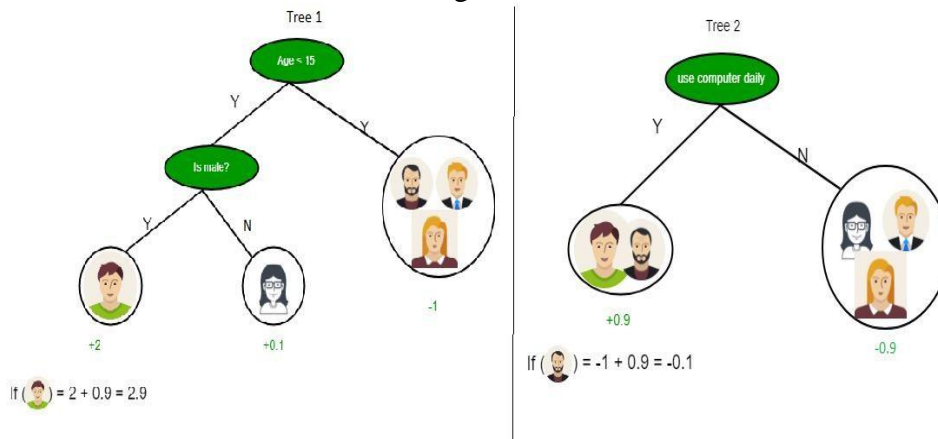
Ans. Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.

Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

We can represent any boolean function on discrete attributes using the decision tree.

Below are some assumptions that we made while using decision tree:

- At the beginning, we consider the whole training set as the root.
- Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- On the basis of attribute values records are distributed recursively.
- We use statistical methods for ordering attributes as root or the internal node.



Definition: Entropy is the measures of **impurity**, **disorder** or **uncertainty** in a bunch of examples. Entropy controls how a Decision Tree decides to **split** the data. It actually effects how a **Decision Tree** draws its boundaries.

The Equation of Entropy:

$$Entropy = - \sum p(X) \log p(X)$$

here p(x) is a fraction of
examples in a given class

Definition: Information gain (IG) measures how much “information” a feature gives us about the class

- **Information gain** is the main key that is used by **Decision Tree Algorithms** to construct a Decision Tree.
- **Decision Trees** algorithm will always tries to maximize **Information gain**.
- An **attribute** with highest **Information gain** will tested/split first.

The Equation of Information gain:

$$\text{Information gain} = \text{entropy (parent)} - [\text{weightes average}] * \text{entropy (children)}$$

Building Decision Tree using Information Gain

The essentials:

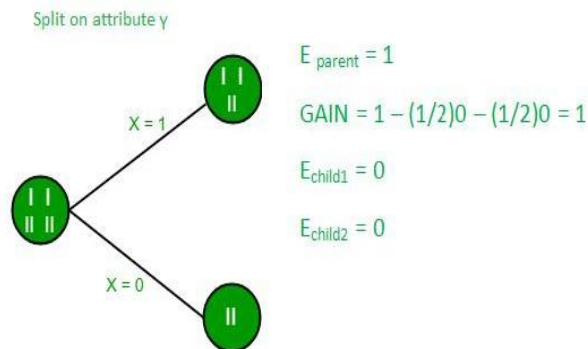
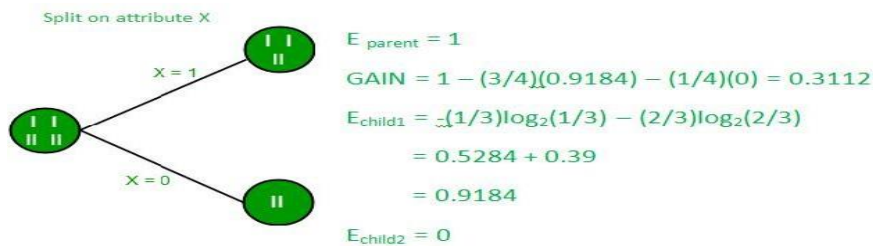
- Start with all training instances associated with the root node
- Use info gain to choose which attribute to label each node with
- **Note:** No root-to-leaf path should contain the same discrete attribute twice
- Recursively construct each subtree on the subset of training instances that would be classified down that path in the tree.
 - **Example:**

- Training set: 3 features and 2 classes

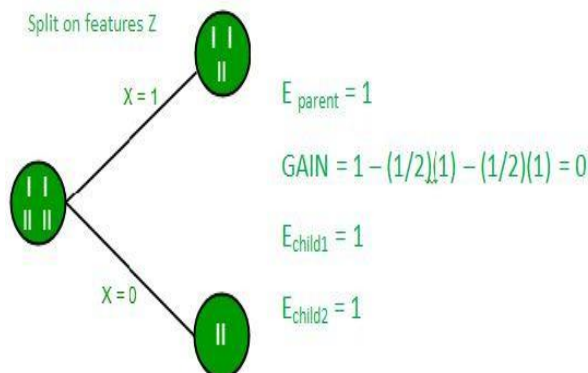
X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Here, we have 3 features and 2 output classes.

To build a decision tree using Information gain. We will take each of the feature and calculate the information for each feature.



Split on feature Y



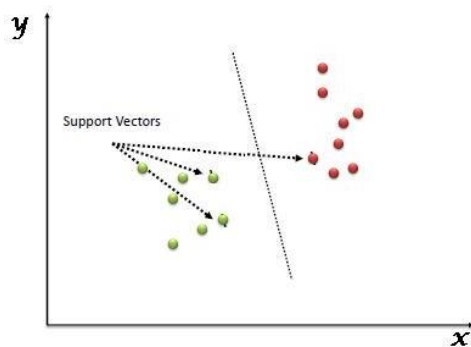
Split on feature Z

From the above images we can see that the information gain is maximum when we make a split on feature Y. So, for the root node best suited feature is feature Y. Now we can see that while splitting the dataset by feature Y, the child contains pure subset of the target variable. So we don't need to further split the dataset.

The final tree for the above dataset would be look like this:

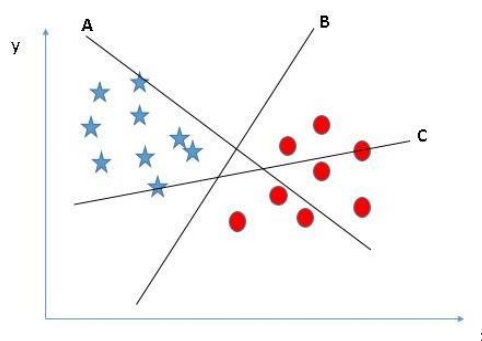
Q2.Describe the working behavior of support vector machine with diagram.

Ans. “Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot).



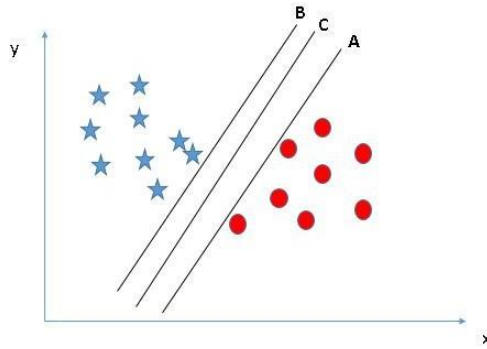
Identify the right hyper-plane (Scenario-1): Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.

You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.



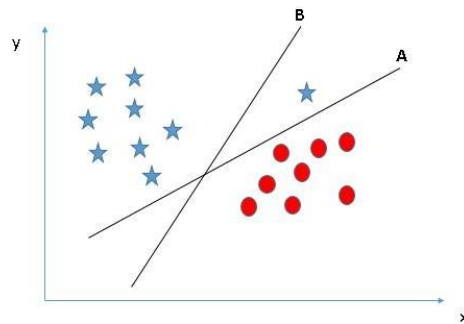
Support Vectors are simply the co-ordinates of individual observation. The SVM classifier is a frontier which best segregates the two classes (hyper-plane/ line).

Identify the right hyper-plane (Scenario-2): Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?



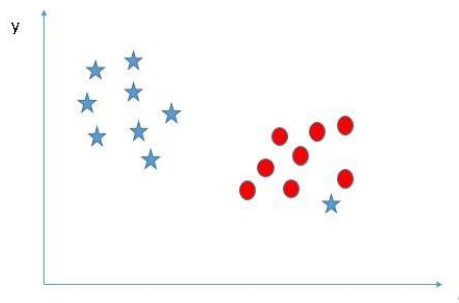
Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**.

- **Identify the right hyper-plane (Scenario-3):**Hint: Use the rules as discussed in previous section to identify the right hyper-plane

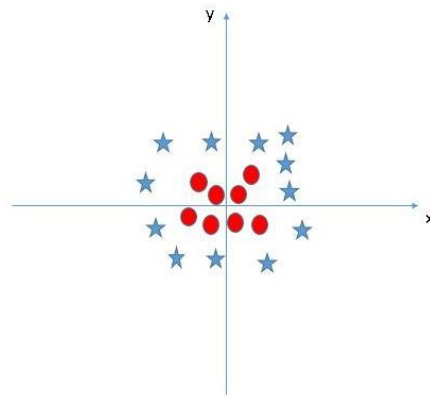


Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A**. But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A**.

Can we classify two classes (Scenario-4)?: Below, I am unable to segregate the two classes using a straight line, as one of the stars lies in the territory of other(circle) class as an outlier.



Find the hyper-plane to segregate to classes (Scenario-5): In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.



SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature $z = x^2 + y^2$. Now, let's plot the data points on axis x and z:

In above plot, points to consider are:

- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

Q3. Discuss Markov Chain Monte Carlo methods in detail.

Ans. Monte Carlo methods learn from complete sample returns

A. Only defined for episodic tasks

Monte Carlo methods learn directly from experience

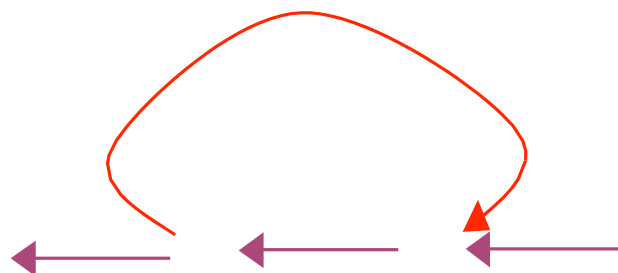
A. On-line: No model necessary and still attains optimality Simulated: No need for a full model

Monte Carlo Policy Evaluation:-

Goal: learn $V(s)$

Given: some number of episodes under which contain s

Idea: Average returns observed after visits to s



Every-Visit MC: average returns for every time s is visited in an episode

First-visit MC: average returns only for first time s is visited in an episode

Both converge asymptotically



First- Visit Monte Carlo Policy Evaluation:-

Initialize:

$\pi \leftarrow$ policy to be evaluated

$V \leftarrow$ an arbitrary state-value function

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

(a) Generate an episode using π

(b) For each state s appearing in the episode:

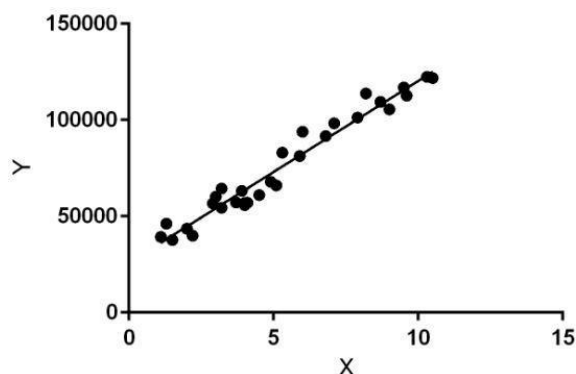
$R \leftarrow$ return following the first occurrence of s

Append R to $Returns(s)$

$V(s) \leftarrow \text{average}(Returns(s))$

Q4.Discuss linear regression with an example.

Ans. Linear Regression is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.



Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Hypothesis function for Linear Regression :

$$y = \theta_1 + \theta_2 \cdot x$$

While training the model we are given :

x: input training data (univariate – one input variable(parameter))

y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ_1 and θ_2 values.

θ_1 : intercept

θ_2 : coefficient of x

Once we find the best θ_1 and θ_2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

Cost Function (J):

By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ_1 and θ_2 values, to reach the best value that minimize the error between predicted y value (pred) and true y value (y).

$$J = \frac{1}{n} \sum_{i=1}^n (pred_i - y_i)^2 \quad \text{minimize} \frac{1}{n} \sum_{i=1}^n (pred_i - y_i)^2$$

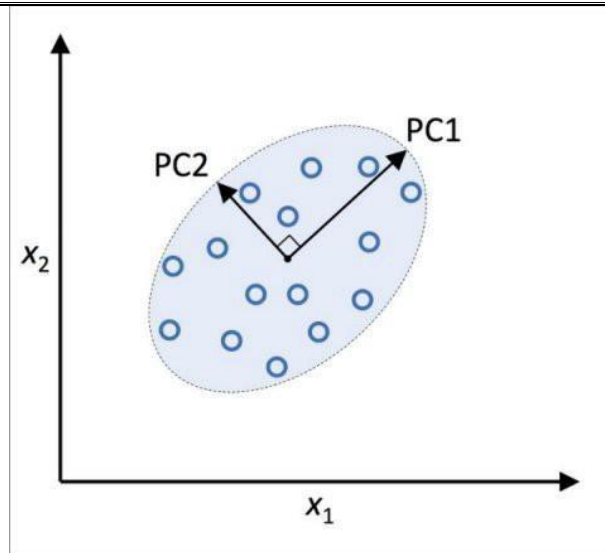
Cost function(J) of Linear Regression is the **Root Mean Squared Error (RMSE)** between predicted y value (pred) and true y value (y).

Q5.How Principal Component Analysis is carried out to reduce dimensionality of data sets?

Ans. Principal Component Analysis (PCA) is an unsupervised linear transformation technique that is widely used across different fields, most prominently for feature extraction and dimensionality reduction. Other popular applications of PCA include exploratory data analyses and de-noising of signals in stock market trading, and the analysis of genome data and gene expression levels in the field of bioinformatics.

PCA helps us to identify patterns in data based on the correlation between features. In a nutshell, PCA aims to find the directions of maximum variance in high-dimensional data and projects it onto a new subspace with equal or fewer dimensions than the original one.

The orthogonal axes (**principal components**) of the new subspace can be interpreted as the directions of maximum variance given the constraint that the new feature axes are orthogonal to each other, as illustrated in the following figure:



In the preceding figure, x_1 and x_2 are the original feature axes, and **PC1** and **PC2** are the principal components. If we use PCA for dimensionality reduction, we construct a $d \times k$ -dimensional transformation matrix \mathbf{W} that allows us to map a sample vector \mathbf{x} onto a new k -dimensional feature subspace that has fewer dimensions than the original d -dimensional feature space:

$$\begin{aligned} \mathbf{x} &= [x_1, x_2, \dots, x_d], & \mathbf{x} &\in \mathbb{R}^d \\ &\downarrow \mathbf{x}\mathbf{W}, & \mathbf{W} &\in \mathbb{R}^{d \times k} \\ \mathbf{z} &= [z_1, z_2, \dots, z_k], & \mathbf{z} &\in \mathbb{R}^k \end{aligned}$$

As a result of transforming the original d -dimensional data onto this new k -dimensional subspace (typically $k \ll d$), the first principal component will have the largest possible variance, and all consequent principal components will have the largest variance given the constraint that these components are uncorrelated (orthogonal) to the other principal components — even if the input features are correlated, the resulting principal components will be mutually orthogonal (uncorrelated).

let's summarize the approach in a few simple steps:

1. Standardize the d -dimensional dataset.
2. Construct the covariance matrix.
3. Decompose the covariance matrix into its eigenvectors and eigenvalues.
4. Sort the eigenvalues by decreasing order to rank the corresponding eigenvectors.
5. Select k eigenvectors which correspond to the k largest eigenvalues, where k is the dimensionality of the new feature subspace ($k \leq d$).
6. Construct a projection matrix \mathbf{W} from the “top” k eigenvectors.
7. Transform the d -dimensional input dataset \mathbf{X} using the projection matrix \mathbf{W} to obtain the new k -dimensional feature subspace.

MODEL PAPER - 2

PART-A

Q1. What are Unsupervised Machine Learning techniques?

Ans. Unsupervised learning is a machine learning technique, where you do not need to supervise the model. Instead, you need to allow the model to work on its own to discover information. It mainly deals with the unlabelled data.

Q2. What is the difference between Supervised and Unsupervised Machine Learning?

Ans.

Basic for comparison	Supervised ml	Unsupervised ml
Basic	Deals with labelled data.	Handles unlabeled data
Computational complexity	High	Low
Analysis	Offline	Real-time
Accuracy	Produces accurate results	Generates moderate results

Q3. What is 'naive' in the Naive Bayes Classifier?

Ans. It is a **classification** technique based on **Bayes'** Theorem with an assumption of independence among predictors. In simple terms, a **Naive Bayes classifier** assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Q4. Compare K-means and KNN Algorithms.

Ans.

	k-NN	k-Means
Type	Supervised	Unsupervised
Meaning of k	Number of closest neighbors to look at	Number of centroids
Calculation of prediction error	Yes	No
Optimization done using	Cross validation, and confusion matrix	Elbow method, silhouette method
Convergence	When all observations classified at the desired accuracy	When cluster memberships don't change anymore
Complexity	Train: $O(d)$ Test: $O(nd)$ Where: d: Dimensions/features n: Number of observations	$O(nkld)$ Where: n: Number of points k: Number of clusters l: Number of iterations d: Number of attributes

Q5. Explain how a system can play a game of chess using Reinforcement Learning.

Ans. Neuro Chess learns chess board evaluation functions, represented by artificial neural networks. It integrates inductive neural network learning, temporal differencing, and a variant of explanation- based learning.

Q6. How will you know which machine learning algorithm to choose for your classification problem?

Ans. Some problems are very specific and require a unique approach. E.g. if you look at a recommender system, it's a very common type of machine learning algorithm and it solves a very specific kind of problem. While some other problems are very open and need a trial & error approach. Supervised learning, classification and regression etc. are very open. They could be used in anomaly detection, or they could be used to build more general sorts of predictive models. Besides some of the decisions that we make when choosing a machine learning algorithm have less to do with the optimization or the technical aspects of the algorithm but more to do with business decisions. Below we look at some of the factors that can help you narrow down the search for your machine learning algorithm.

Q7. How is Amazon able to recommend other things to buy? How does the recommendation engine work?

Ans. Amazon uses recommendations as a targeted marketing tool in both email campaigns and on most of its web sites' pages. Here are the different ways they are currently using recommendations:

Existing recommendation algorithms couldn't scale to Amazon's tens of millions of customers and products, so they decided to develop their own.

Amazon currently uses item-to-item collaborative filtering, which scales to massive data sets and produces high-quality recommendations in real time.

This type of filtering matches each of the user's purchased and rated items to similar items, then combines those similar items into a recommendation list for the user.

Their recommendation algorithm is an effective way of creating a personalized shopping experience for each customer which helps Amazon increase average order value and the amount of revenue generated from each customer.

Q8. When will you use classification over regression?

Ans. Classification is used when the output variable is a category such as "red" or "blue", "spam" or "not spam". It is used to draw a conclusion from observed values. Differently from, regression which is used when the output variable is a real or continuous value like "age", "salary", etc. When we must identify the class, the data belongs to we use classification over regression. Like when you must identify whether a name is male or female instead of finding out how they are correlated with the person.

Q9. What are the three stages of building a model in machine learning?

Ans. a) Model building :- Model building is a hobby that involves the creation of models either from kits or from materials and components acquired by the builder.

b) Model testing :- Model-based testing is an application of model-based design for designing and optionally also executing artifacts to perform software testing or system testing. Models can be used to represent the desired behavior of a system under test, or to represent testing strategies and a test environment.

c) Applying the model : Model-based testing allows large numbers of test cases to be generated from a description of the behavior of the system under test. Given the same description and test runner, many types of scenarios can be exercised and large areas of the application under test can be covered, thus leading to a more effective and more efficient testing process.

Q10. What is the difference between Inductive Machine Learning and Deductive Machine Learning?

Sol.

	Inductive Reasoning	Deductive Reasoning
DEFINITION	A logical process where multiple premises that are assumed to be true are combined to obtain a specific conclusion	A logical process where a conclusion is based on the concordance of multiple premises that are generally assumed to be true
PREMISE AND CONCLUSION	Moves from specific premises to a general conclusion	Moves from general premises to a specific conclusion
VALIDITY OF CONCLUSION	Conclusions may be incorrect even if the argument is strong and the premises are true	Conclusions can be proven valid if the premises are true
APPROACH	Bottom-up approach	Top-Down approach
BASE	Based on patterns and connections	Based on facts, truths or rules

PART-B

Q1. What are important objectives of machine learning? What are the basic design issues and approaches to machine learning?

Sol. Objective:- The primary purpose of **machine learning** is to discover patterns in the user data and then make predictions based on these and intricate patterns for answering business questions and solving business problems. Machine learning helps in analysing the data as well as identifying trends.

Basic design Issue :--

Data Quality :

1. The most common issue when using ML is poor data quality. The adage is true: garbage in, garbage out. It is essential to have good quality data to produce quality ML algorithms and models. To get high-quality data, you must implement data evaluation, integration, exploration, and governance techniques prior to developing ML models.
2. Common issues include lack of good clean data, the ability to apply the correct learning algorithms, black-box approach, the bias in training data/algorithms, etc. Another issue we see is model maintenance. When you think about traditional and coded software, it becomes more and more stable over time, and as you detect bugs, you are able to make tweaks to fix it and make it better. With ML being optimized towards the outcomes, self-running and dependent on the underlying data process, there can

be some model degradation that might lead to less optimal outcomes. Assuming ML will work faultlessly postproduction is a mistake and we need to be laser-focused on monitoring the ML performance post-deployment as well.

Transparency :

The most common issue I find to be is the lack of model transparency. It is often very difficult to make definitive statements on how well a model is going to generalize in new environments. You have to often ask, “what are the modes of failure and how do we fix them.”

As with any AI/ML deployment, the “one-size-fits-all” notion does not apply and there is no magical “out of the box” solution. Specific products and scenarios will require specialized supervision and custom fine-tuning of tools and techniques. Additionally, assuming ML models use unsupervised and closed-loop techniques, the goal is that the tooling will auto-detect and self-correct. However, we have found AI/ML models can be biased. Sometimes the system may be more conservative in trying to optimize for error handling, error correction, in which case the performance of the product can take a hit. The tendency for certain conservative algorithms to over-correct on specific aspects of the SDLC is an area where organizations will need to have better supervision.

Manpower :

Having data and being able to use it so does not introduce bias into the model. How organizations change how they think about software development and how they collect and use data. Make sure they have enough skill sets in the organization. More software developers are coming out of school with ML knowledge. Provide the opportunity to plan and prototype ideas.

Approaches :- The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

Supervised learning

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal.

Unsupervised learning

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data.

Semi – supervised learning

Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Some of the training examples are missing training labels, yet many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy.

Reinforcement learning

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms.

Q2. Describe inbrief:

Sol. (a) Hypothesis space search in decision:--- The *hypothesis space* used by a machine learning system is the set of all hypotheses that might possibly be returned by it. It is typically defined by a Hypothesis Language, possibly in conjunction with a Language Bias.

Hypothesis Space Search by ID3

- ID3 searches the space of possible decision trees: doing hill-climbing on information gain.
- It searches the *complete* space of all finite discrete-valued functions. All functions have at least one tree that represents them.
- It maintains only one hypothesis (unlike Candidate-Elimination). It cannot tell us how many other viable ones there are.
- It does not do back tracking. Can get stuck in local optima.
- Uses all training examples at each step. Results are less sensitive to errors.

(b) Inductive bias in decision:--

The **inductive bias** (also known as **learning bias**) of a learning algorithm is the set of assumptions that the learner uses to predict outputs given inputs that it has not encountered. In machine learning, one aims to construct algorithms that are able to *learn* to predict a certain target output. To achieve this, the learning algorithm is presented some training examples that demonstrate the intended relation of input and output values. Then the learner is supposed to approximate the correct output, even for examples that have not been shown during training. Without any additional assumptions, this problem cannot be solved exactly since unseen situations might have an arbitrary output value. The kind of necessary assumptions about the nature of the target function are subsumed in the phrase *inductive bias*.

Q3. Describe k-nearest neighbor algorithm .Why is it called instance based learning?

Sol. “Nearest-neighbor” learning is also known as “Instance-based” learning.

K-Nearest Neighbours, or KNN, is a family of simple:

- classification
- and regression algorithms

Based on Similarity (Distance) calculation between instances.

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

K-Nearest Neighbours

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

The KNN Algorithm:

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data
 - 3.1 Calculate the distance between the query example and the current example from the data.
 - 3.2 Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels.

Choosing the right value for K

To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before.

Here are some things to keep in mind:

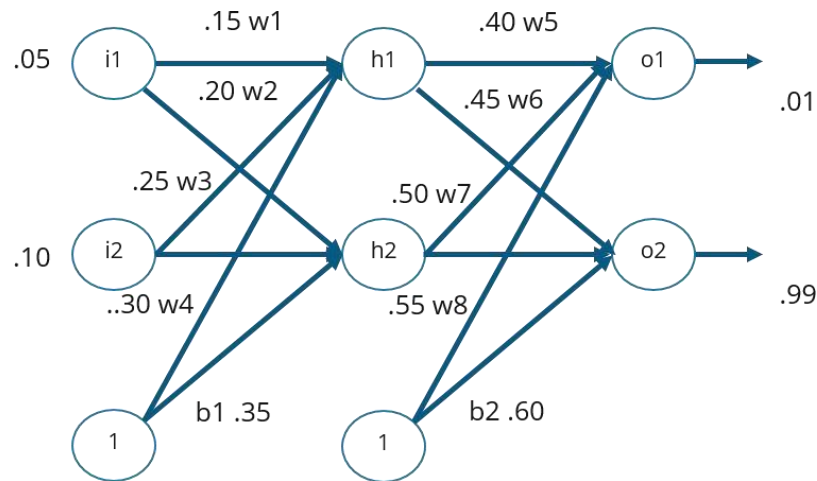
1. As we decrease the value of K to 1, our predictions become less stable. Just think for a minute, imagine $K=1$ and we have a query point surrounded by several reds and one green (I'm thinking about the top left corner of the colored plot above), but the green is the single nearest neighbor. Reasonably, we would think the query point is most likely red, but because $K=1$, KNN incorrectly predicts that the query point is green.
2. Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.
3. In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

Ques 4. What are the steps in Back propagation algorithm? Why a Multilayer neural network is required?

Ans. Back propagation : The Backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function is then considered to be a solution to the learning problem.

How Backpropagation Works?

Consider the below Neural Network:



The above network contains the following:

- two inputs
- two hidden neurons
- two output neurons
- two biases

Below are the steps involved in Backpropagation:

- Step – 1: Forward Propagation
- Step – 2: Backward Propagation
- Step – 3: Putting all the values together and calculating the updated weight value

Backpropagation Algorithm:

initialize network weights (often small random values)

do

forEach training example named ex

 prediction = neural-net-output(network, ex) // *forward pass*

 actual = teacher-output(ex)

 compute error (prediction - actual) at the output units

compute $\{\Delta w_{\{h\}}\}$ for all weights from hidden layer to output layer // *backward pass*

compute $\{\Delta w_{\{i\}}\}$ for all weights from input layer to hidden layer // *backward pass*

continued

 update network weights // *input layer not modified by error estimate*

until all examples classified correctly or another stopping criterion satisfied

return the network

Q5. Write a short note on Support Vector machine.

Sol. In machine learning, support-vector machines (SVMs, also support-vector networks[1]) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space,

mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.

New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high dimensional feature spaces.

When data are unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The **support-vector clustering**[2] algorithm, created by Hava Siegelmann and Vladimir Vapnik, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data, and is one of the most widely used clustering algorithms in industrial applications.

A support vector machine is a supervised learning algorithm that sorts data into two categories. It is trained with a series of data already classified into two categories, building the model as it is initially trained. The task of an SVM algorithm is to determine which category a new data point belongs in. This makes SVM a kind of non-binary linear classifier.

An SVM algorithm should not only place objects into categories, but have the margins between them on a graph as wide as possible.

Some applications of SVM include:

- Text and hyper text classification
- Image classification
- Recognizing hand written characters
- Biological sciences, including protein classification.

Q6. Explain policy iteration and value iteration in brief.

Sol. Value Iteration

Value iteration computes the optimal state value function by iteratively improving the estimate of $V(s)$. The algorithm initialize $V(s)$ to arbitrary random values. It repeatedly updates the $Q(s, a)$ and $V(s)$ values until they converges. Value iteration is guaranteed to converge to the optimal values. This algorithm is shown in the following pseudo-code:

```
Initialize  $V(s)$  to arbitrary values
Repeat
  For all  $s \in S$ 
    For all  $a \in \mathcal{A}$ 
       $Q(s, a) \leftarrow E[r|s, a] + \gamma \sum_{s' \in S} P(s'|s, a)V(s')$ 
     $V(s) \leftarrow \max_a Q(s, a)$ 
Until  $V(s)$  converge
```

Policy iteration :

While value-iteration algorithm keeps improving the value function at each iteration until the value-function converges. Since the agent only cares about the finding the optimal policy, sometimes the optimal policy will converge before the value function. Therefore, another algorithm called policy-iteration instead of repeated improving the value-function estimate, it will re-define the policy at each step and compute the value according to this new policy until the policy converges. Policy iteration is also guaranteed to converge to the optimal policy and it often takes less iterations to converge than the value-iteration algorithm. The pseudo code for Policy Iteration is shown below.

```
Initialize a policy  $\pi'$  arbitrarily
Repeat
     $\pi \leftarrow \pi'$ 
    Compute the values using  $\pi$  by
        solving the linear equations
        
$$V^\pi(s) = E[r|s, \pi(s)] + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) V^\pi(s')$$

    Improve the policy at each state
        
$$\pi'(s) \leftarrow \arg \max_a (E[r|s, a] + \gamma \sum_{s' \in S} P(s'|s, a) V^\pi(s'))$$

Until  $\pi = \pi'$ 
```

Q7. Which machine learning algorithm can you use for better analysis service to earn maximum profit? Explain with example.

Sol. Support Vector Machine is a supervised machine learning algorithm for classification or regression problems where the dataset teaches SVM about the classes so that SVM can classify any new data. It works by classifying the data into different classes by finding a line (hyperplane) which separates the training data set into classes. As there are many such linear hyperplanes, SVM algorithm tries to maximize the distance between the various classes that are involved and this is referred as margin maximization. If the line that maximizes the distance between the classes is identified, the probability to generalize well to unseen data is increased.

SVM's are classified into two categories:

- Linear SVM's – In linear SVM's the training data i.e. classifiers are separated by a hyperplane.
- Non-Linear SVM's- In non-linear SVM's it is not possible to separate the training data using a hyperplane. For example, the training data for Face detection consists of group of images that are faces and another group of images that are not faces (in other words all other images in the world except faces). Under such conditions, the training data is too complex that it is impossible to find a representation for every feature vector. Separating the set of faces linearly from the set of non-face is a complex task.

Advantages of Using SVM

- SVM offers best classification performance (accuracy) on the training data.
- SVM renders more efficiency for correct classification of the future data.
- The best thing about SVM is that it does not make any strong assumptions on data.
- It does not over-fit the data.

Applications of Support Vector Machine

- SVM is commonly used for stock market forecasting by various financial institutions. For instance, it can be used to compare the relative performance of the stocks when compared to performance of other stocks in the same sector. The relative comparison of stocks helps manage investment making decisions based on the classifications made by the SVM learning algorithm.
- Data Science Libraries in Python to implement Support Vector Machine –SciKit Learn, PyML , SVM^{Struct} Python , LIBSVM
- Data Science Libraries in R to implement Support Vector Machine – klar, e1071

PART-C

Q1. Provide outline of Decision tree from the training tuples. Also list down the different attribute selection measure used in the decision tree construction.

Ans. Decision Tree Analysis is a general, predictive modelling tool that has applications spanning a number of different areas. In general, decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

The decision rules are generally in form of if-then-else statements. The deeper the tree, the more complex the rules and fitter the model.

Before we dive deep, let's get familiar with some of the terminologies:

- (1) Instances: Refer to the vector of features or attributes that define the input space
- (2) Attribute: A quantity describing an instance
- (3) Concept: The function that maps input to output
- (4) Target Concept: The function that we are trying to find, i.e., the actual answer
- (5) Hypothesis Class: Set of all the possible functions
- (6) Sample: A set of inputs paired with a label, which is the correct output (also known as the Training Set)
- (7) Candidate Concept: A concept which we think is the target concept
- (8) Testing Set: Similar to the training set and is used to test the candidate concept and determine its performance

Decision trees are highly favoured classifiers because of the resemblance of their understandable nature to the branched process of human thinking. But the comprehensible rationality of these trees can be severely affected by the bias in the selection of the split attribute, and the traditional heuristic methods appear to be multi-value.

. Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.

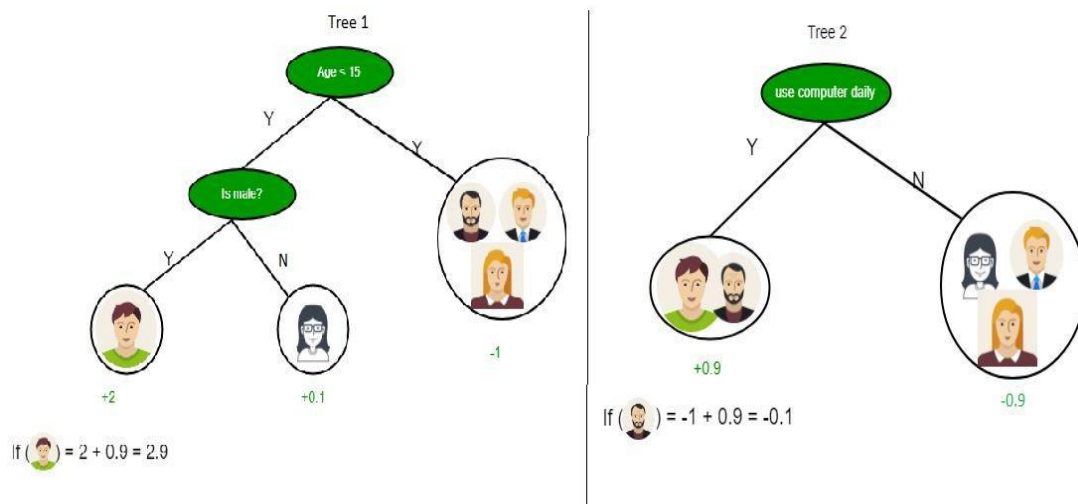
- Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

We can represent any boolean function on discrete attributes using the decision tree.

Below are some assumptions that we made while using decision tree:

At the beginning, we consider the whole training set as the root.

- Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- On the basis of attribute values records are distributed recursively.
- We use statistical methods for ordering attributes as root or the internal node



Definition: Entropy is the measures of **impurity, disorder** or **uncertainty** in a bunch of examples. Entropy controls how a Decision Tree decides to **split** the data. It actually effects how a **Decision Tree** draws its boundaries.

The Equation of Entropy:

$$Entropy = - \sum p(X) \log p(X)$$

here p(x) is a fraction of examples in a given class

Definition: Information gain (IG) measures how much “information” a feature gives us about the class

- Information gain** is the main key that is used by **Decision Tree Algorithms** to construct a Decision Tree.
- Decision Trees** algorithm will always tries to maximize **Information gain**.
- An **attribute** with highest **Information gain** will tested/split first.

The Equation of Information gain:

$$\text{Information gain} = \text{entropy (parent)} - [\text{weightes average}] * \text{entropy (children)}$$

Building Decision Tree using Information Gain

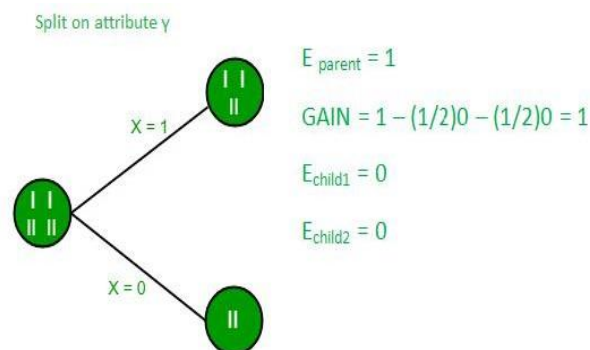
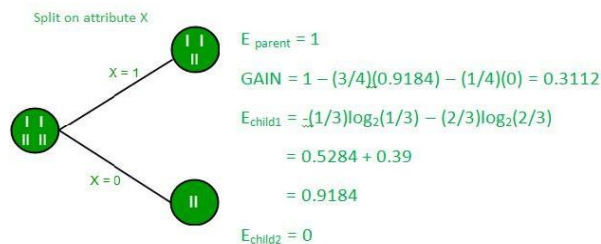
The essentials:

- Start with all training instances associated with the root node
- Use info gain to choose which attribute to label each node with
- *Note:* No root-to-leaf path should contain the same discrete attribute twice
- Recursively construct each subtree on the subset of training instances that would be classified down that path in the tree.
 - **Example:**
 - **Training set: 3 features and 2 classes**

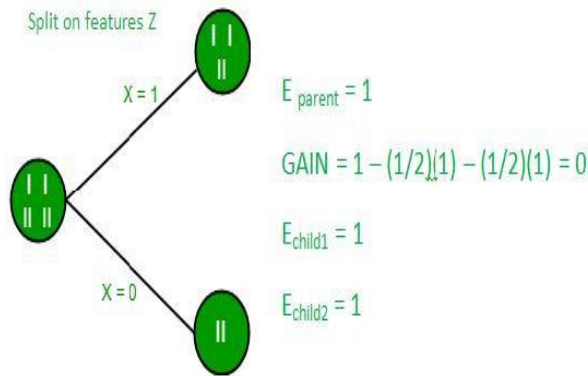
X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Here, we have 3 features and 2 output classes.

To build a decision tree using Information gain. We will take each of the feature and calculate the information for each feature.



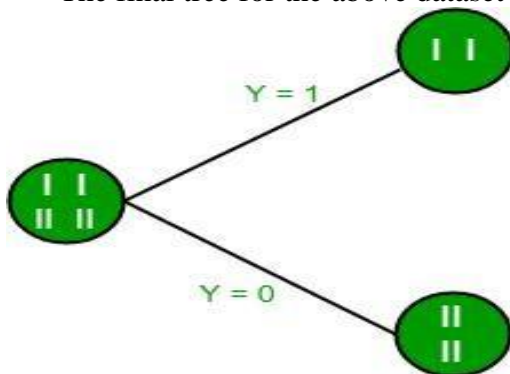
Split on feature Y



Split on feature Z

From the above images we can see that the information gain is maximum when we make a split on feature Y. So, for the root node best suited feature is feature Y. Now we can see that while splitting the dataset by feature Y, the child contains pure subset of the target variable. So we don't need to further split the dataset.

The final tree for the above dataset would be look like this:



Q2. Elaborate various issues like control learning control policy, Q-learning and convergence in reinforcement learning.

Sol. Control learning:

Many control problems encountered in areas such as robotics and automated driving require complex, nonlinear control architectures. Techniques such as gain scheduling, robust control, and nonlinear model predictive control (MPC) can be used for these problems, but often require significant domain expertise from the control engineer. For example, gains and parameters are difficult to tune. The resulting controllers can pose implementation challenges, such as the computational intensity of nonlinear MPC.

You can use deep neural networks, trained using reinforcement learning, to implement such complex controllers. These systems can be self-taught without intervention from an expert control engineer. Also, once the system is trained, you can deploy the reinforcement learning policy in a computationally efficient way.

You can also use reinforcement learning to create an end-to-end controller that generates actions directly from raw data, such as images. This approach is attractive for video-intensive

Applications, such as automated driving, since you do not have to manually define and select image features.

A control task in RL is where the policy is not fixed, and the goal is to find the optimal policy. That is, to find the policy $\pi(a|s)\pi(a|s)$ that maximises the expected total reward from any given state.

A control algorithm based on value functions (of which Monte Carlo Control is one example) usually works by also solving the prediction problem, i.e. it predicts the values of acting in different ways, and adjusts the policy to choose the best actions at each step.

Prediction is described as the computation of $v_{\pi}(s)$ and $q_{\pi}(s,a)$ for a fixed arbitrary policy π , where

- $v_{\pi}(s)$ is the value of a state s under policy π , given a set of episodes obtained by following π and passing through s .
- $q_{\pi}(s,a)$ is the action-value for a state-action pair (s,a) . It's the expected return when starting in state s , taking action a , and thereafter following policy π .

Control is described as approximating optimal policies. When doing control, one maintains both an approximate policy and an approximate value function. The value function is repeatedly altered to more closely approximate the value function for the current policy, and the policy is repeatedly improved with respect to the current value function. This is the idea of generalised policy iteration (GPI)

Control policy :

The H_{∞} control design problem is considered for nonlinear systems with unknown internal system model. It is known that the nonlinear H_{∞} control problem can be transformed into solving the so-called Hamilton-Jacobi-Isaacs (HJI) equation, which is a nonlinear partial differential equation that is generally impossible to be solved analytically. Even worse, model-based approaches cannot be used for approximately solving HJI equation, when the accurate system model is unavailable or costly to obtain in practice. To overcome these difficulties, an off-policy reinforcement learning (RL) method is introduced to learn the solution of HJI equation from real system data instead of mathematical system model, and its convergence is proved. In the off-policy RL method, the system

data can be generated with arbitrary policies rather than the evaluating policy, which is extremely important and promising for practical systems. For implementation purpose, a neural network (NN)-based actor-critic structure is employed and a least-square NN weight update algorithm is derived based on the method of weighted residuals. Finally, the developed NN-based off-policy RL method is tested on a linear F16 aircraft plant, and further applied to a rotational/translational actuator system.

Q-learning >

Q-learning is a model-free reinforcement learning algorithm to learn a policy telling an agent what action to take under what circumstances. It does not require a model (hence the connotation "model-free") of the environment, and it can handle problems with stochastic transitions and rewards, without requiring adaptations.

For any finite Markov decision process (FMDP), Q-learning finds an optimal policy in the sense maximizing the expected value of the total reward over any and all successive steps, starting from the current state.[1] Q-learning can identify an optimal action-selection policy for any given FMDP, given infinite exploration time and a partly-random policy.[1]"Q" names the function that returns the reward used to provide the reinforcement and can be said to stand for the "quality" of an action taken in a given state.

Create a q-table

When q-learning is performed we create what's called a q-table or matrix that follows the shape of [state, action] and we initialize our values to zero. We then update and store our

q-values after an episode. This q-table becomes a reference table for our agent to select the best action based on the q-value.

Q-learning and making updates

The next step is simply for the agent to interact with the environment and make updates to the state action pairs in our q-table Q [state, action].

Taking Action: Explore or Exploit

An agent interacts with the environment in 1 of 2 ways. The first is to use the q-table as a reference and view all possible actions for a given state. The agent then selects the action based on the max value of those actions. This is known as **exploiting** since we use the information we have available to us to make a decision.

The second way to take action is to act randomly. This is called **exploring**. Instead of selecting actions based on the max future reward we select an action at random. Acting randomly is important because it allows the agent to explore and discover new states that otherwise may not be selected during the exploitation process. You can balance exploration/exploitation using epsilon (ϵ) and setting the value of how often you want to explore vs. exploit. Here's some rough code that will depend on how the state and action space are setup.

Convergence:

This paper examines the convergence of pay off and strategies in Erev and Roth's model of reinforcement learning. When all players use this rule it eliminates iteratively dominated strategies and in two-person constant-sum games average pay offs converge to the value of the game. Strategies converge in constant-sum games with unique equilibria if they are pure or if they are mixed and the game is 2×2 . The long-run behaviour of the learning rule is governed by equations related to Maynard Smith's version of replicator dynamic. Properties of the learning rule against general opponents are also studied.

Under the real options approach to investment under uncertainty, agents formulate optimal policies under the assumption that firms' growth prospects do not vary over time. This paper proposes and solves a model of investment decisions in which the growth rate and volatility of the decision variable shift between different states at random times. A value-maximizing investment policy is derived such that in each regime the firm's investment policy is optimal and recognizes the possibility of a regime shift. Under this policy, investment is intermittent and increases with marginal q . Moreover, investment typically is very small but, in some states, the capital stock jumps. Implications for marginal q and the user cost of capital are also examined.

Q3. State all the parameters of Markov Decision model in detail.

Sol. The Markov decision process (MDP) is a mathematical framework for sequential decision making under uncertainty that has informed decision making in a variety of application areas including inventory control, scheduling, finance, and medicine (Puterman 1994, Boucherie and Van Dijk 2017). MDPs generalize Markov chains in that a decision maker (DM) can take actions to influence the rewards and transition dynamics of the system. When the transition dynamics and rewards are known with certainty, standard dynamic programming methods can be used to find an optimal policy, or set of decisions, that will maximize the expected rewards over the planning horizon.

Unfortunately, the estimates of rewards and transition dynamics used to parameterize the MDPs are often imprecise and lead the DM to make decisions that do not perform well with respect to the true system. The imprecision in the estimates arises because these values are typically

obtained from observational data or from multiple external sources. When the policy found via an optimization process using the estimates is evaluated under the true parameters, the performance can be much worse than anticipated (Mannor et al. 2007). This motivates the need for MDPs that account for this ambiguity in the MDP parameters.

In this article, we are motivated by situations in which the DM relies on external sources to parameterize the model but has multiple credible choices which provide potentially conflicting estimates of the parameters. In this situation, the DM may be grappling with the following questions: Which source should be used to parameterize the model? What are the potential implications of using one source over another? To address these questions, we propose a new method that allow the DM to simultaneously consider multiple models of the MDP parameters and create a policy that balances the performance while being no more complicated than an optimal policy for an

MDP that only considers one model of the parameters.

1. Markov decision processes

MDPs are a common framework for modeling sequential decision making that influences a stochastic reward process. For ease of explanation, we introduce the MDP as an interaction between an exogenous actor, nature, and the DM. The sequence of events that define the MDP are as follows: first, nature randomly selects an initial state $s_1 \in S$ according to the initial distribution $\mu_1 \in M(S)$, where $M(\cdot)$ denotes the set of probability measures on the discrete set. The DM observes the state $s_1 \in S$ and selects an action $a_1 \in A$. Then, the DM receives a reward $r_1(s_1, a_1) \in R$ and then nature selects a new state $s_2 \in S$ with probability $p_1(s_2 | s_1, a_1) \in [0, 1]$. This process continues whereby for any decision epoch $t \in T \equiv \{1, \dots, T\}$, the DM observes the state $s_t \in S$, selects an action $a_t \in A$, and receives a reward $r_t(s_t, a_t)$, and nature selects a new state $s_{t+1} \in S$ with probability $p_t(s_{t+1} | s_t, a_t)$. The DM selects the last action at time T which may influence which

state is observed at time $T + 1$ through the transition probabilities. Upon reaching $s_{T+1} \in S$ at time $T + 1$, the DM receives a terminal reward of $r_{T+1}(s_{T+1}) \in R$. Future rewards are discounted at a rate of $\alpha \in (0, 1]$ which accounts for the preference of rewards received now over rewards received in the future. In this article, we assume without loss of generality that the discount factor is already incorporated into the reward definition. We will refer to the times at which the DM selects an action as the set of decision epochs, T , the set of rewards as $R \in R|S \times A \times T|$, and the set of transition

probabilities as $P \in \mathbb{R}^{S \times A \times S \times T}$ with elements satisfying $p_t(s_{t+1} | s_t, a_t) \in [0, 1]$ and $\sum_{s_{t+1} \in S} p_t(s_{t+1} | s_t, a_t) = 1, \forall t \in T, s_t \in S, a_t \in A$. Throughout the remainder of this article, we will use the tuple (T, S, A, R, P, γ) to summarize the parameters of an MDP.

The realized value of the DM's sequence of actions is the total reward over the planning horizon:

$$\sum_{t=1}^T r_t(s_t, a_t) + \gamma V(s_{T+1}).$$

The objective of the DM is to select the sequence of actions in a strategic way so that the expectation of (1) is maximized. Thus, the DM will select the actions at each decision epoch based on some information available to her. The strategy by which the DM selects the action for each state at decision epoch $t \in T$ is called a decision rule, $\pi_t \in \Pi_t$, and the set of decision rules over the planning horizon is called a policy, $\pi \in \Pi$. There exist two dichotomies in the classes of policies that a DM may select from: 1) history- dependent vs. Markov, and 2) randomized vs. deterministic. History-dependent policies may consider the entire history of the MDP, $h_t = (s_1, a_1, \dots, a_{t-1}, s_t)$, when prescribing which action to select at decision epoch $t \in T$, while Markov policies only consider the current state $s_t \in S$ when selecting an action. Randomized policies specify a probability distribution over the action set, $\pi_t(s_t) \in \mathcal{M}(A)$, such that action $a_t \in A$ will be selected with probability $\pi_t(a_t | s_t)$.

Deterministic policies specify a single action to be selected with probability 1. Markov policies are a subset of history-dependent policies, and deterministic policies are a subset of randomized policies.

Q4. Explain the steps in K-means clustering algorithm. Cluster the following set of 4 object Into two cluster using K-means A(3,5), B(4,5), C(1,3), D(2,4). Consider object C as the initial cluster center.

Sol. K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as bary center of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more. Finally, this algorithm aims at minimizing an objective function known as squared error function given by:

$$J(V) = \sum_{i=1}^c \sum_{j=1}^{c_i} (\|x_i - v_j\|)^2$$

where,

' $\|x_i - v_j\|$ ' is the Euclidean distance between x_i and v_j .

' c_i ' is the number of data points in i^{th} cluster.

' c ' is the number of cluster centers.

Algorithmic steps for k-means clustering

Let $X = \{x_1, x_2, x_3, \dots, x_n\}$ be the set of data points and $V = \{v_1, v_2, \dots, v_c\}$ be the set of centers.

- 1) Randomly select ' c ' cluster centers.
- 2) Calculate the distance between each data point and cluster centers.
- 3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers..
- 4) Recalculate the new cluster center using:

$$v_i = (1 / c_i) \sum_{j=1}^{c_i} x_i$$

where, ' c_i ' represents the number of data points in i^{th} cluster.

- 5) Recalculate the distance between each data point and new obtained cluster centers.
- 6) If no data point was reassigned then stop, otherwise repeat from step 3).

Advantages

- 1) Fast, robust and easier to understand.
- 2) Relatively efficient: $O(tknd)$, where n is # objects, k is # clusters, d is # dimension of each object, and t is # iterations. Normally, $k, t, d \ll n$.
- 3) Gives best result when data set are distinct or well separated from each other.

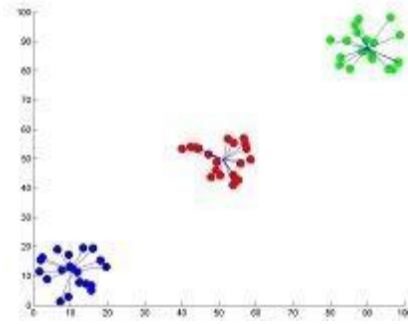


Fig I: Showing the result of k-means for ' N ' = 60 and ' c ' = 3

Disadvantages

- 1) The learning algorithm requires apriori specification of the number of cluster centers.
- 2) The use of Exclusive Assignment - If there are two highly overlapping data then k-means will not be able to resolve that there are two clusters.
- 3) The learning algorithm is not invariant to non-linear transformations i.e. with different representation of data we get different results (data represented in form of cartesian co-ordinates and polar co-ordinates will give different results).
- 4) Euclidean distance measures can unequally weight underlying factors.
- 5) The learning algorithm provides the local optima of the squared error function.
- 6) Randomly choosing of the cluster center cannot lead us to the fruitful result. Pl. refer Fig.
- 7) Applicable only when mean is defined i.e. fails for categorical data.
- 8) Unable to handle noisy data and outliers.
- 9) Algorithm fails for non-linear data set.

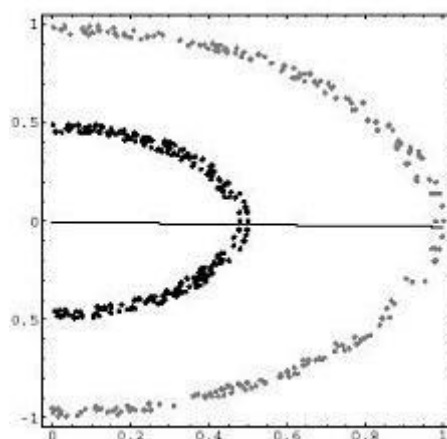


Fig II: Showing the non-linear data set where k-means algorithm fails

Q5. Elaborate on the types of machine learning and discuss the components in design of a learning system.

Sol.At a high-level, machine learning is simply the study of teaching a computer program or algorithm how to progressively improve upon a set task that it is given. On the research-side of things, machine learning can be viewed through the lens of theoretical and mathematical modeling of how this process works. However, more practically it is the study of how to build applications that exhibit this iterative improvement. There are many ways to frame this idea, but largely there are three major recognized categories: supervised learning, unsupervised learning, and reinforcement learning.

Supervised Learning:

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

Supervised learning classified into two categories of algorithms:

Classification: A classification problem is when the output variable is a category, such as “Red” or “blue” or “disease” and “no disease”.

Regression: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Unsupervised learning

Unsupervised learning is the training of machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data.

Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore machine is restricted to find the hidden structure in unlabeled data by our-self.

Unsupervised learning classified into two categories of algorithms:

Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.

Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Reinforcement Learning

Reinforcement learning is fairly different when compared to supervised and unsupervised learning. Where we can easily see the relationship between supervised and unsupervised (the presence or absence of labels), the relationship to reinforcement learning is a bit murkier.

For any reinforcement learning problem, we need an agent and an environment as well as a way to connect the two through a feedback loop. To connect the agent to the environment,

we give it a set of actions that it can take that affect the environment. To connect the environment to the agent, we have it continually issue two signals to the agent: an updated state and a reward (our reinforcement signal for behavior).

Where is reinforcement learning in the real world?

Video Games: One of the most common places to look at reinforcement learning is in learning to play games. Look at Google's reinforcement learning application, AlphaZero and AlphaGo which learned to play the game Go.

Industrial Simulation: For many robotic applications (think assembly lines), it is useful to have our machines learn to complete their tasks without having to hardcode their processes. This can be a cheaper and safer option; it can even be less prone to failure.

Resource Management: Reinforcement learning is good for navigating complex environments. It can handle the need to balance certain requirements. Take, for example, Google's data centers. They used reinforcement learning to balance the need to satisfy our power requirements, but do it as efficiently as possible, cutting major costs.

Design of a learning system :

Data Model selection Learning Application or Testing

1. Data: $D = \{d_1, d_2, \dots, d_n\}$

2. Model selection:

- Select a model or a set of models (with parameters)

E.g. $y = ax + b + \varepsilon, \varepsilon = N(0, \sigma)$

- Select the error function to be optimized

$$\frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

E.g.

Learning:

- Find the set of parameters optimizing the error function

– The model and parameters with the smallest error

4. Application (Evaluation):

- Apply the learned model

– E.g. predict y s for new inputs x using learned $f(x)$.

MODEL PAPER - 3

PART-A

Q1. What are the application of supervised machine learning in modern business?

Ans=> Supervised learning is the approach in data science so widely . In the past few years, machine learning (ML) has revolutionized the way we do business. Please note that the survey covered both supervised and unsupervised .Recommendation sections are something we can't imagine modern.

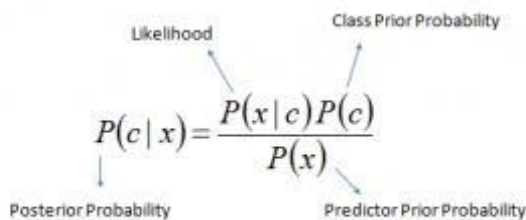
Q2.What is semi supervised machine learning?

Ans=>Semi-supervised learning is an approach to machine learning that combines a small amount of labeled data with a large amount of unlabeled data during training. Semi-supervised learning falls between unsupervised learning (with no labeled training data) and supervised learning (with only labeled training data).

Q3.What is 'naïve' in the naïve bayes classifier?

Ans=> It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$. Look at the equation below:



The diagram shows the equation $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$ with arrows pointing from labels to the corresponding parts of the equation. 'Likelihood' points to $P(x|c)$, 'Class Prior Probability' points to $P(c)$, 'Posterior Probability' points to $P(c|x)$, and 'Predictor Prior Probability' points to $P(x)$.

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Q4.When will you use classification over regression?

Ans=>Classification is about identifying group membership while regression technique involves predicting a response. Classification technique is preferred over regression when the results of the model need to return the belongingness of data points in a dataset to specific explicit categories.

Q5.What is Deep Learning?

Ans=> Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

Q6.Compare k-means and KNN algorithms?

Ans=>K-means is a clustering algorithm that tries to partition a set of points into K sets (clusters) such that the points in each cluster tend to be near each other. It is unsupervised because the points have no external classification.

K-nearest neighbors is a classification (or regression) algorithm that in order to determine the classification of a point, combines the classification of the K nearest points. It is supervised because you are trying to classify a point based on the known classification of other points.

Q7. What is Randon forest?

Ans=> Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

Q8.What is Bias and variance in a machine learning model?

Ans=>Bias is the simplifying assumptions made by the model to make the target function easier to approximate. Variance is the amount that the estimate of the target function will change given different training data.

Q9.What is the trade-off between Bias and variance?

Ans=> Bias is the simplifying assumptions made by the model to make the target function easier to approximate. Variance is the amount that the estimate of the target function will change given different training data. Trade-off is tension between the error introduced by the bias and the variance.

Q10.Define precision and recall ?

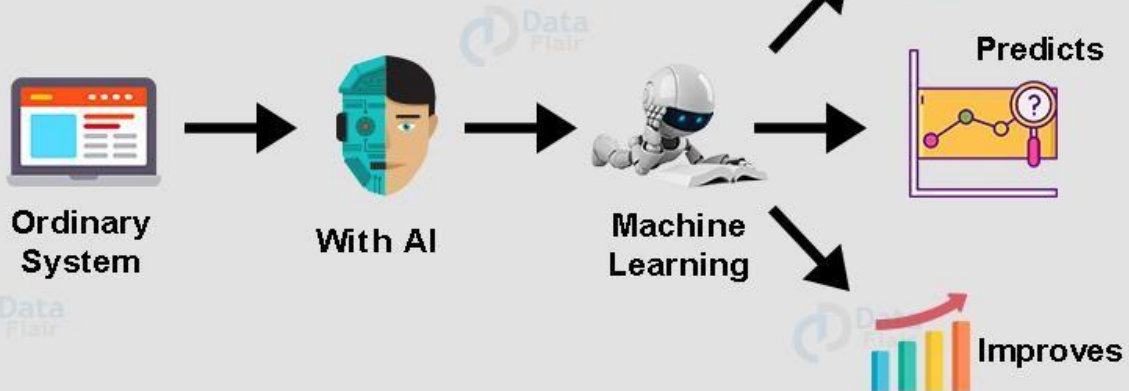
Ans=> Precision and recall are two extremely important model evaluation metrics. While precision refers to the percentage of your results which are relevant.

recall refers to the percentage of total relevant results correctly classified by your algorithm.

PART-B**Q2.What is machine learning? Explain different perspective and issues in machine.**

Ans=>Machine Learning is the most popular technique of predicting the future or classifying information to help people in making necessary decisions. Machine Learning algorithms are trained over instances or examples through which they learn from past experiences and also analyze the historical data. Therefore, as it trains over the examples, again and again, it is able to identify patterns in order to make predictions about the future.

Introduction to Machine Learning



Machine Learning Algorithms can be classified into 3 types as follows –

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Supervised learning

In Supervised Learning, the dataset on which we train our model is labeled. There is a clear and distinct mapping of input and output. Based on the example inputs, the model is able to get trained in the instances. An example of supervised learning is spam filtering. Based on the labeled data, the model is able to determine if the data is spam or ham. This is an easier form of training. Spam filtering is an example of this type of machine learning algorithm

Unsupervised Learning

In Unsupervised Learning, there is no labeled data. The algorithm identifies the patterns within the dataset and learns them. The algorithm groups the data into various clusters based on their density. Using it, one can perform visualization on high dimensional data. One example of this type of Machine learning algorithm is the Principle Component Analysis. Furthermore, K-Means Clustering is another type of Unsupervised Learning where the data is clustered in groups of a similar order.

The learning process in Unsupervised Learning is solely on the basis of finding patterns in the data. After learning the patterns, the model then makes conclusions.

Reinforcement Learning

Reinforcement Learning is an emerging and most popular type of Machine Learning Algorithm. It is used in various autonomous systems like cars and industrial robotics. The aim of this algorithm is to reach a goal in a dynamic environment. It can reach this goal based on several rewards that are provided to it by the system.

It is most heavily used in programming robots to perform autonomous actions. It is also used in making intelligent self-driving cars. Let us consider the case of robotic navigation. Furthermore, the efficiency can be improved with further experimentation with the agent in its environment. This the main principle behind reinforcement learning. There are similar sequences of action in a reinforcement learning model.

Issues of ml:-

- Focusing Too Much on Algorithms and Theories. ...
- Mastering ALL of ML. ...
- Using Changing or Premade Tools. ...
- Having Algorithms Become Obsolete as Soon as Data Grows. ...
- Getting Bad Predictions to Come Together With Biases. ...
- Making the Wrong Assumptions. ...
- Receiving Bad Recommendations. ...
- Having Bad Data Convert to Bad Results.

Q3.What is Artificial NEURAL NETWORK, Explain appropriate problem for neural.

Ans=> Artificial neural networks (ANN) or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains.^[1] Such systems "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge of cats, for example, that they have fur, tails, whiskers and cat-like faces. Instead, they automatically generate identifying characteristics from the examples that they process.

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it.

Problem:

1. Neural networks are supposed to be able to mimic any continuous function.
2. But many a times we are stuck with networks not performing up to the mark, or it takes a whole lot of time to get decent results.
3. One should approach the problem statistically rather than going with gut feelings regarding the changes which should be brought about in the architecture of the network.
4. One of the first steps should be proper preprocessing of data.
5. Other than mean normalisation and scaling, Principal Component Analysis may be useful in speeding up training. If the dimension of the data is reduced to such an extent that a proper amount of variance is still retained, one can save on space without compromising much on the quality of the data. Also, neural networks can be trained faster when they are provided with less data.

Q4.Explain the concept of a perception with a neat diagram and represent the Boolean function of AND, OR using perception?

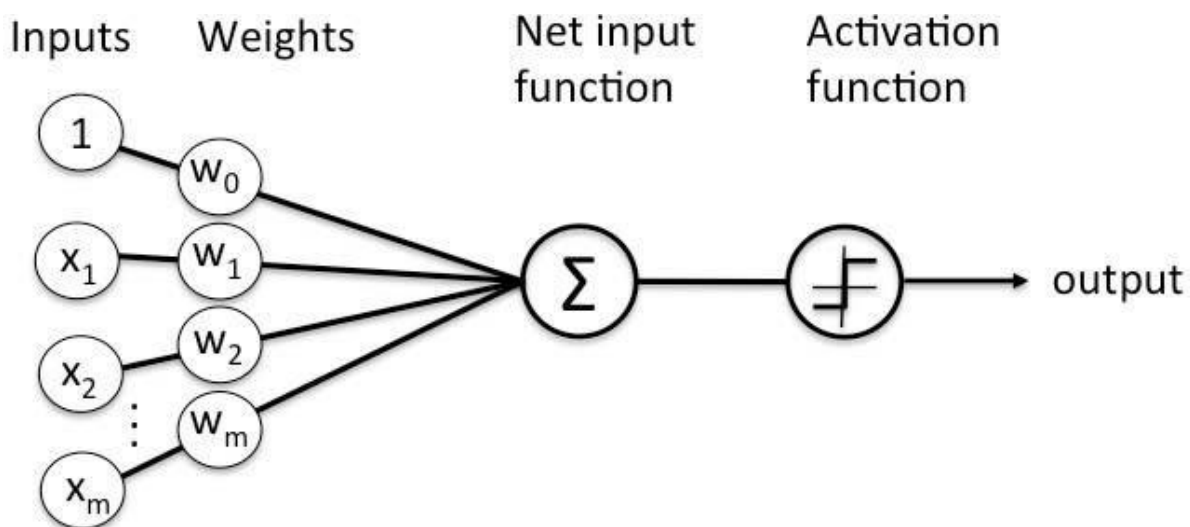
Ans=> Perception (from the Latin perceptio) is the organization, identification, and interpretation of sensory information in order to represent and understand the presented information or environment.^[2]

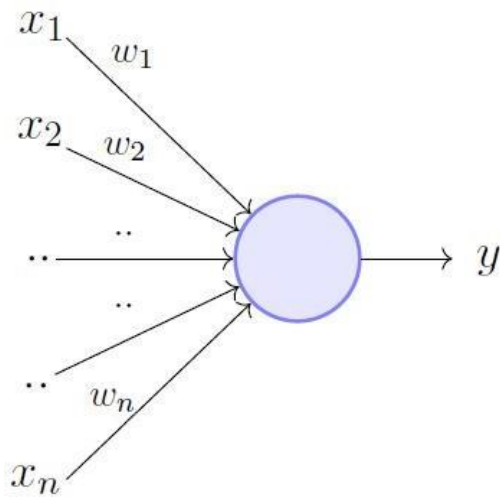
All perception involves signals that go through the nervous system, which in turn result from physical or chemical stimulation of the sensory system.^[3] For example, vision involves light striking the retina of the eye; smell is mediated by odor molecules; and hearing involves pressure waves.

Perception is not only the passive receipt of these signals, but it's also shaped by the recipient's learning, memory, expectation, and attention.^{[4][5]} Sensory input is a process that transforms this low-level information to higher-level information (e.g., extracts shapes for object recognition).^[5] The process that follows connects a person's concepts and expectations (or knowledge), restorative and selective mechanisms (such as attention) that influence perception.

Perception depends on complex functions of the nervous system, but subjectively seems mostly effortless because this processing happens outside conscious awareness.^[3]

Since the rise of experimental psychology in the 19th century, psychology's understanding of perception has progressed by combining a variety of techniques.^[4] Psychophysics quantitatively describes the relationships between the physical qualities of the sensory input and perception.^[6] Sensory neuroscience studies the neural mechanisms underlying perception. Perceptual systems can also be studied computationally, in terms of the information they process. Perceptual issues in philosophy include the extent to which sensory qualities such as sound, smell or color exist in objective reality rather than in the mind of the perceiver.^[1]





$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i \geq \theta$$

$$= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i < \theta$$

Rewriting the above,

$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i - \theta \geq 0$$

$$= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i - \theta < 0$$

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

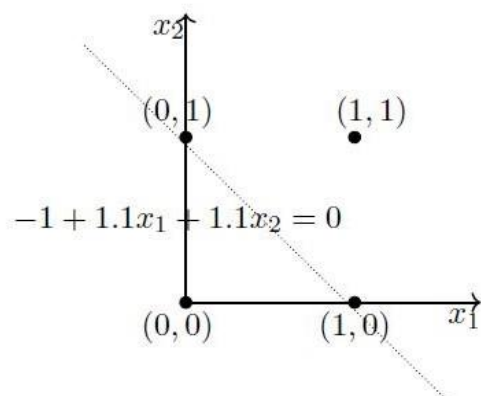
$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 > -w_0$$

One possible solution is

$$w_0 = -1, w_1 = 1.1, w_2 = 1.1$$



Q5 Find the SVD of the matrix $A=[1010 \ 0101]$?

Solution

We first compute $AA^T = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ and $A^T A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$.

We see immediately that the eigenvalues of AA^T are $\lambda_1 = \lambda_2 = 2$ (and hence, the eigenvalues of $A^T A$ are 2 and 0, both with multiplicity 2). Thus, the matrix A has singular value $\sigma_1 = \sigma_2 = \sqrt{2}$.

Next, an orthonormal basis of eigenvectors of AA^T is $u_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $u_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. (You can choose any orthonormal basis of \mathbb{R}^2 here because AA^T is a multiple of the identity, but the one chosen makes computation easiest.) Thus, we set $U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

For $A^T A$, the eigenvectors which correspond to the value of 2 are obtained from the formula below:

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = 2 \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow \begin{array}{l} x + z = 2x \Rightarrow z = x \\ y + w = 2y \Rightarrow y = w \\ x + z = 2z \Rightarrow x = z \\ y + w = 2w \Rightarrow y = w \end{array}$$

Therefore the eigenvectors which correspond to the eigenvalue of 2 are of the form

$$\begin{bmatrix} x \\ y \\ x \\ y \end{bmatrix} = x \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} + y \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}. \text{ Therefore two orthonormal eigenvectors are } \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \text{ and } \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

The eigenvectors which correspond to the eigenvalue of 0 are obtained from the formula below:

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \begin{matrix} x+z=0 \Rightarrow z=-x \\ y+w=0 \Rightarrow w=-y \end{matrix}$$

Therefore the eigenvectors which correspond to the eigenvalue of 0 are of the form

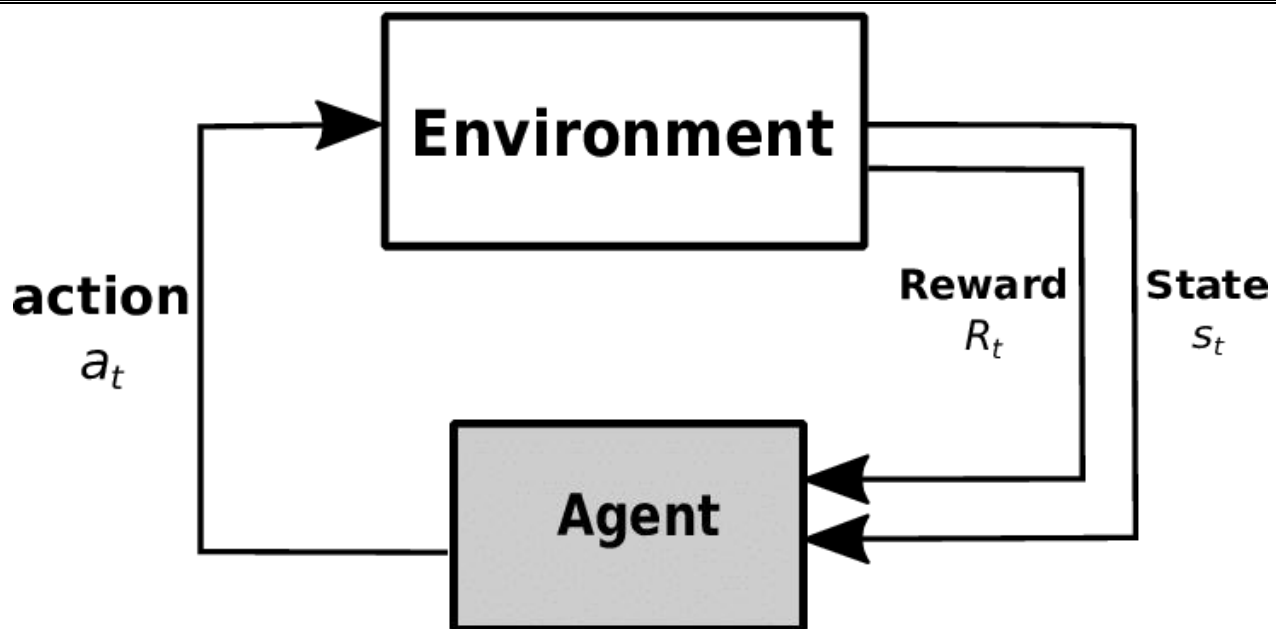
$$\begin{bmatrix} x \\ y \\ -x \\ -y \end{bmatrix} = x \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} + y \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix}. \text{ Therefore two orthonormal eigenvectors are } \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} \text{ and } \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix}.$$

$$\text{Therefore, } V = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 & 0 & 0 \\ 0 & \sqrt{2} & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

Q6.What is reinforcement learning?

Ans=> Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of training dataset, it is bound to learn from its experience.



Main points in Reinforcement learning –

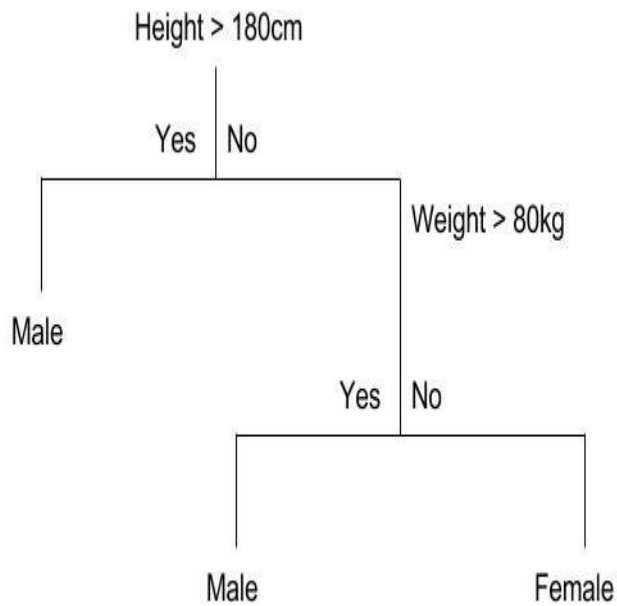
- Input: The input should be an initial state from which the model will start
- Output: There are many possible output as there are variety of solution to a particular problem
- Training: The training is based upon the input, The model will return a state and the user will decide to reward or punish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

PART-C

Q2.Explain the Classification and Regression Tree(CART) with example?

Ans=> The CART or Classification & Regression Trees methodology refers to these two types of decision trees. While there are many classification and regression trees tutorials and classification and regression trees ppts out there, here is a simple definition of the two kinds of decisions trees. It also includes classification and regression trees examples.

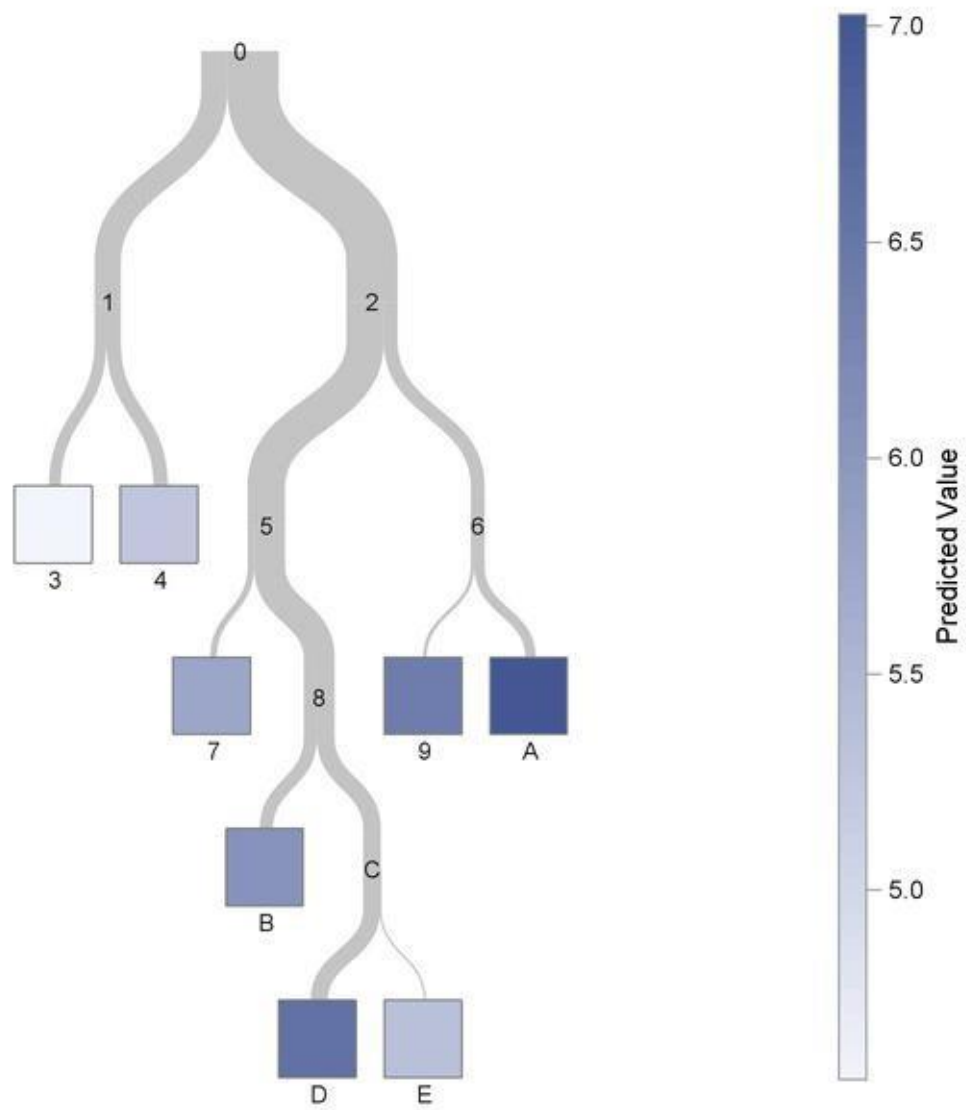
- (i) Classification Trees: A classification tree is an algorithm where the target variable is fixed or categorical. The algorithm is then used to identify the “class” within which a target variable would most likely fall. An example of a classification-type problem would be determining who will or will not subscribe to a digital platform; or who will or will not graduate from high school. These are examples of simple binary classifications where the categorical dependent variable can assume only one of two, mutually exclusive values. In other cases, you might have to predict among a number of different variables. For instance, you may have to predict which type of smartphone a consumer may decide to purchase. In such cases, there are multiple values for the categorical dependent variable. Here’s what a classic classification tree looks like.



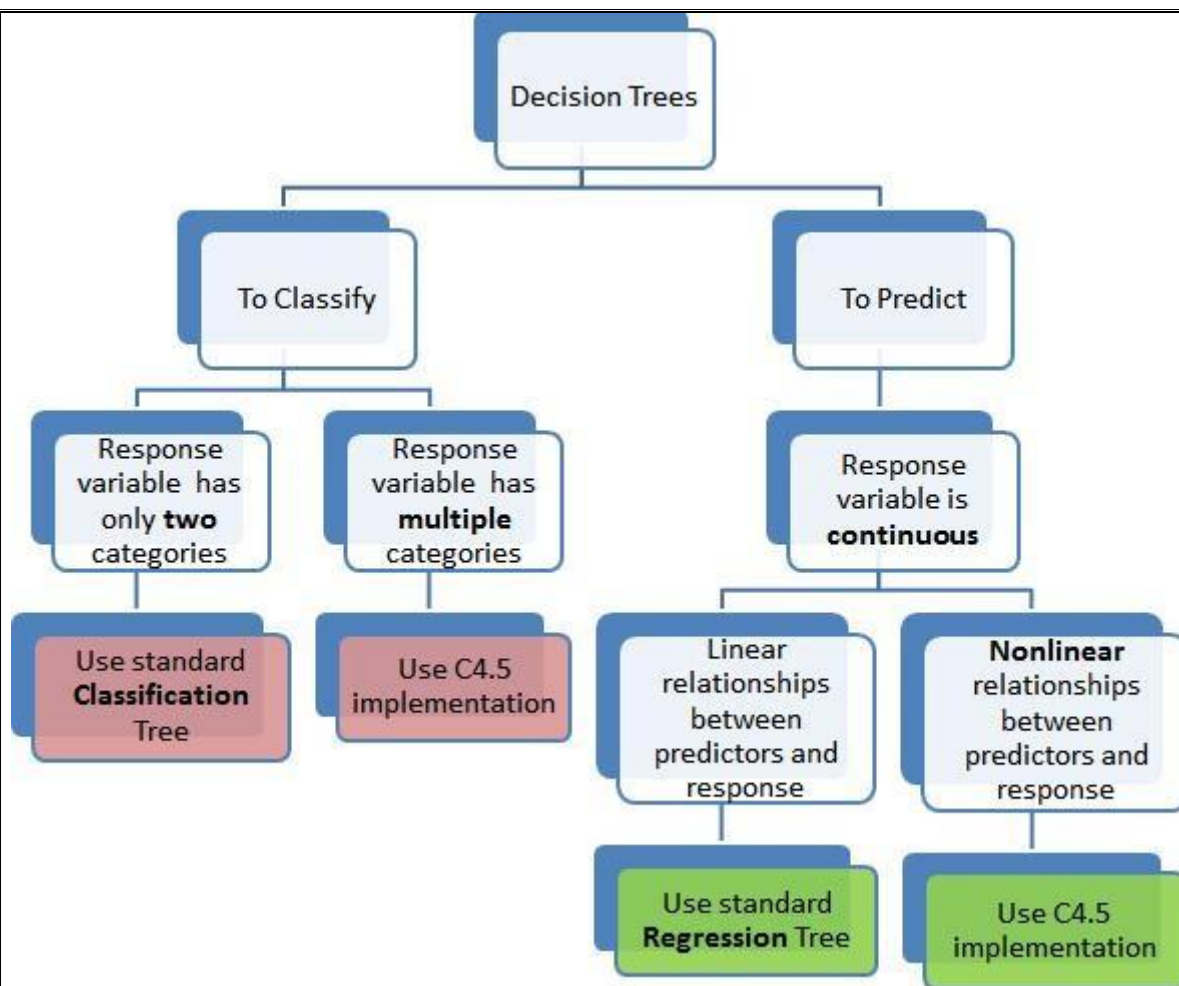
|

- (ii) Regression Trees: A regression tree refers to an algorithm where the target variable is and the algorithm is used to predict it's value. As an example of a regression type problem, you may want to predict the selling prices of a residential house, which is a continuous dependent variable. This will depend on both continuous factors like square footage as well as categorical factors like the style of home, area in which the property is located and so on.
-

Regression Tree for logSalary



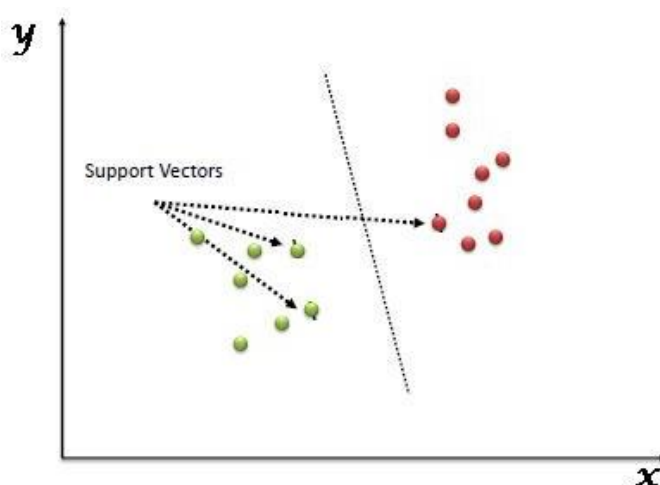
Regression Tree



CART Working

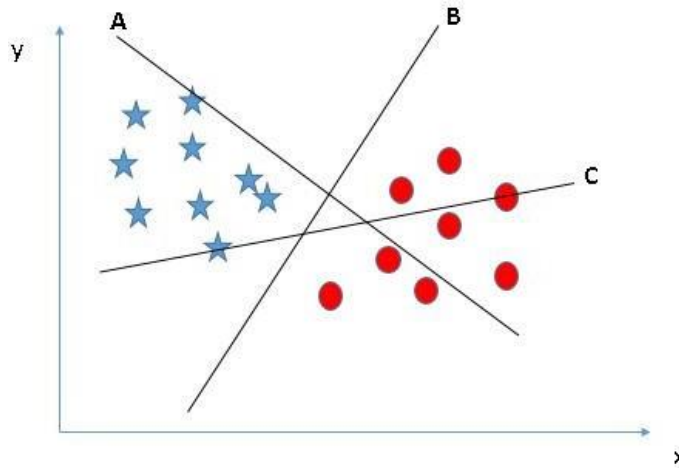
Q3.Describe Support Vector Machine with suitable example?

Ans=> “Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot).

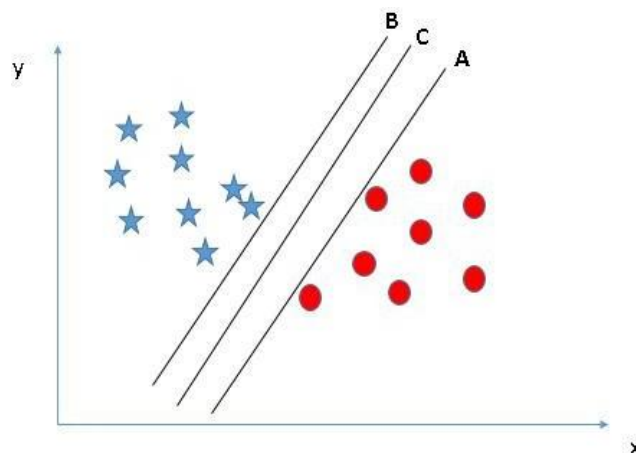


Support Vectors are simply the co-ordinates of individual observation. The SVM classifier is a frontier which best segregates the two classes (hyper-plane/ line).

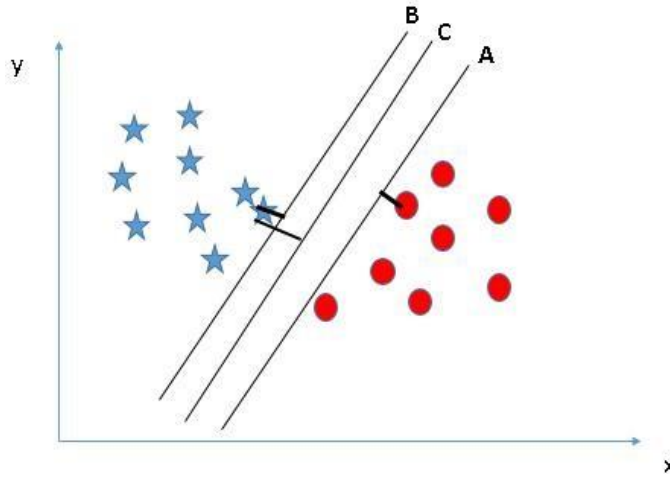
- Identify the right hyper-plane (Scenario-1): Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.



- You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.
- Identify the right hyper-plane (Scenario-2): Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?

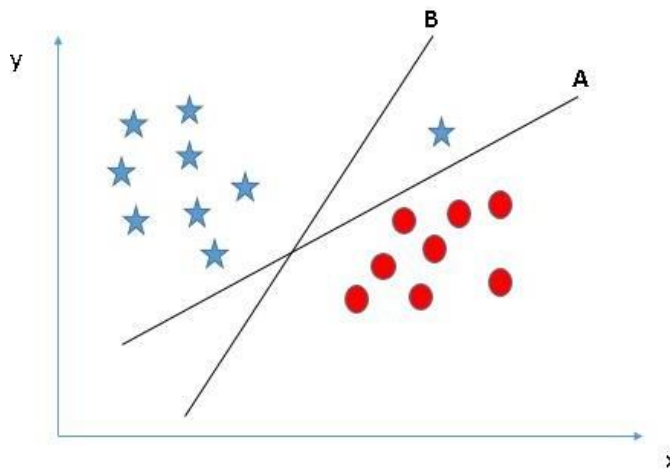


Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin. Let's look at the below snapshot:



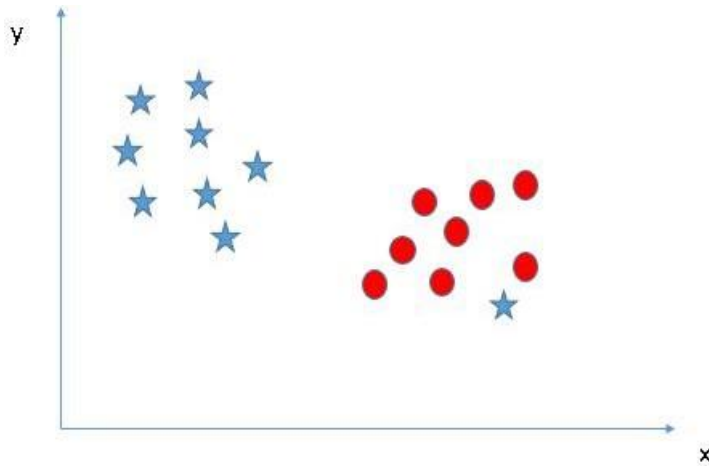
Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

- Identify the right hyper-plane (Scenario-3): Hint: Use the rules as discussed in previous section to identify the right hyper-plane

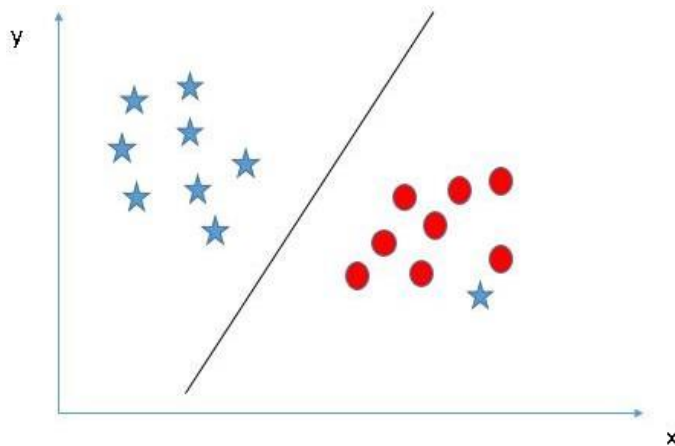


Some of you may have selected the hyper-plane B as it has higher margin compared to A. But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is A.

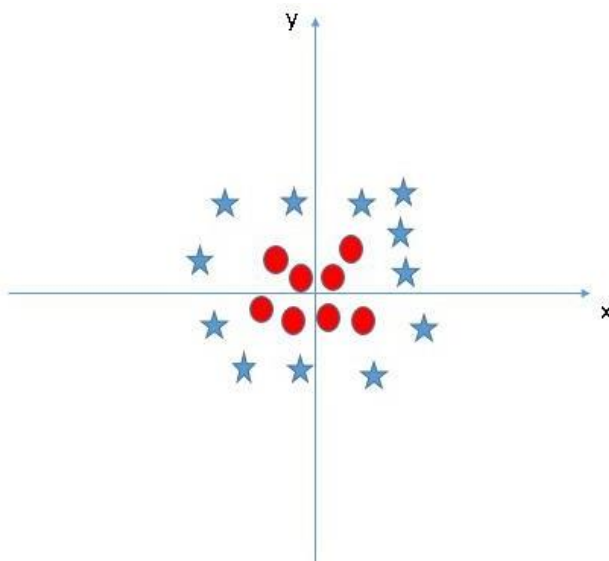
- Can we classify two classes (Scenario-4)? Below, I am unable to segregate the two classes using a straight line, as one of the stars lies in the territory of other(circle) class as an outlier.



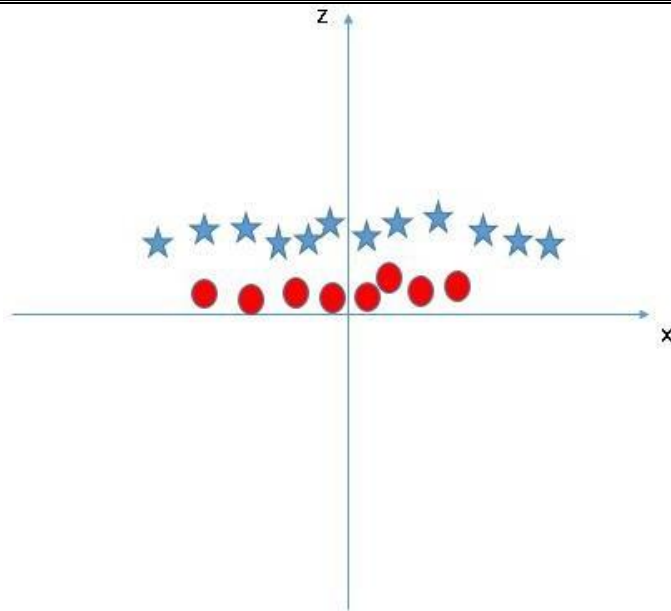
- As I have already mentioned, one star at other end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. Hence, we can say, SVM classification is robust to outliers.



- Find the hyper-plane to segregate to classes (Scenario-5): In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.



- SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature $z = x^2 + y^2$. Now, let's plot the data points on axis x and z:

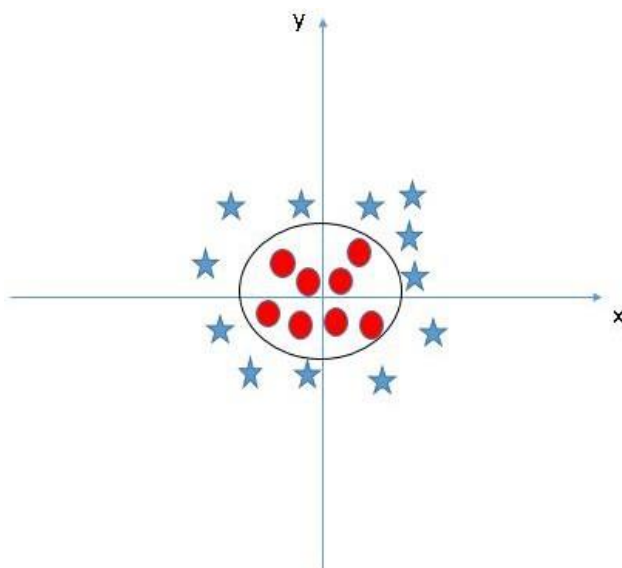


In above plot, points to consider are:

- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z .

In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, the SVM algorithm has a technique called the kernel trick. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e. it converts not separable problem to separable problem. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then finds out the process to separate the data based on the labels or outputs you've defined.

When we look at the hyper-plane in original input space it looks like a circle:



Q4.Describe Principle Component Analysis in detail.Explain how reduce dimension using pca?

Ans=> PCA is mostly used as a tool in exploratory data analysis and for making predictive models. It is often used to visualize genetic distance and relatedness between populations. PCA can be done by eigenvalue decomposition of a data covariance (or correlation) matrix or singular value

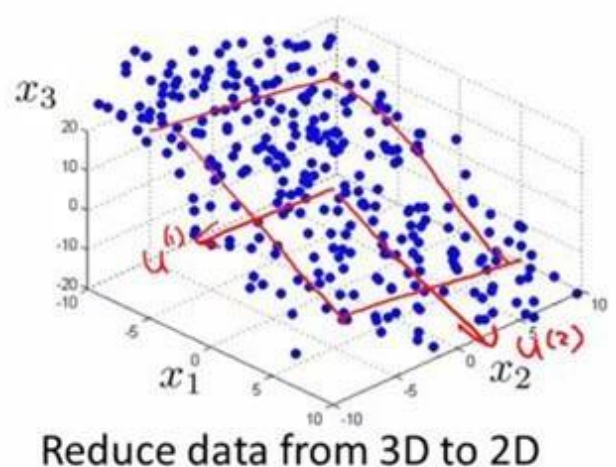
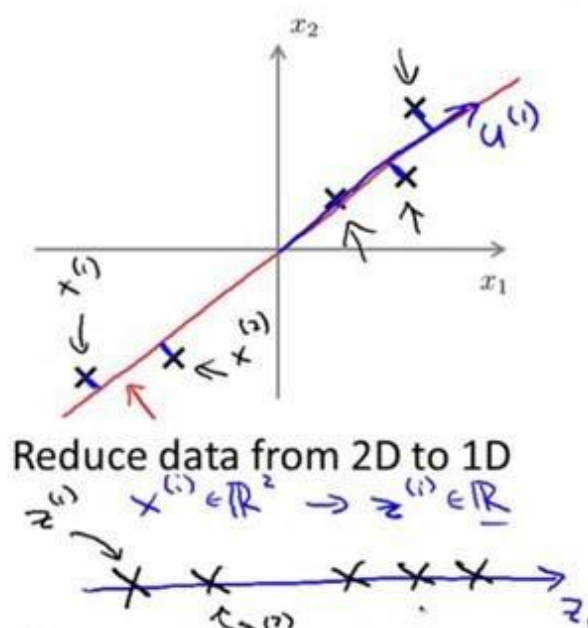
decomposition of a data matrix, usually after a normalization step of the initial data. The normalization of each attribute consists of mean centering – subtracting each data value from its variable's measured mean so that its empirical mean (average) is zero. Some fields, in addition to normalizing the mean, do so for each variable's variance (to make it equal to 1); see z-scores.^[4] The results of a PCA are usually discussed in terms of component scores, sometimes called factor scores (the transformed variable values corresponding to a particular data point), and loadings (the weight by which each standardized original variable should be multiplied to get the component score).^[5] If component scores are standardized to unit variance, loadings must contain the data variance in them (and that is the magnitude of eigenvalues). If component scores are not standardized (therefore they contain the data variance) then loadings must be unit-scaled, ("normalized") and these weights are called eigenvectors; they are the cosines of orthogonal rotation of variables into principal components or back.

The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent. The same is done by transforming the variables to a new set of variables, which are known as the principal components (or simply, the PCs) and are orthogonal, ordered such that the retention of variation present in the original variables decreases as we move down in the order. So, in this way, the 1st principal component retains maximum variation that was present in the original components. The principal components are the eigenvectors of a covariance matrix, and hence they are orthogonal.

Importantly, the dataset on which PCA technique is to be used must be scaled. The results are also sensitive to the relative scaling. As a layman, it is a method of summarizing data. Imagine some wine bottles on a dining table. Each wine is described by its attributes like colour, strength, age, etc. But redundancy will arise because many of them will measure related properties. So what PCA will do in this case is summarize each wine in the stock with less characteristics.

Intuitively, Principal Component Analysis can supply the user with a lower-dimensional picture, a projection or "shadow" of this object when viewed from its most informative viewpoint.

Principal Component Analysis (PCA) algorithm



Algorithm Description:

What is K-means?

1. Partitional clustering approach
 2. Each cluster is associated with a centroid (center point)
 3. Each point is assigned to the cluster with the closest centroid
 4. Number of clusters K must be specified
4. Number of clusters, K, must be specified

Basic Algorithm of K-means

Algorithm Statement:-

Details of K-means:

1. Initial centroids are often chosen randomly. - Clusters produced vary from one run to another
2. The centroid is (typically) the mean of the points in the cluster.
3. 'Closeness' is measured by Euclidean distance, cosine similarity, correlation, etc
4. K-means will converge for common similarity measures mentioned above.
5. Most of the convergence happens in the first few iterations. - Often the stopping condition is changed to 'Until relatively few points change clusters'

' Algorithm Statement

Euclidean Distance

$$d(i, j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{ip} - x_{jp}|^2}$$

MODEL PAPER - 4

PART-A

1. What is bias and variance in a machine learning model?

Ans = Bias is the simplifying assumptions made by the **model** to make the target function easier to approximate. **Variance** is the amount that the estimate of the target function will change given different training data.

2. What are the differences between Machine Learning and Deep Learning?

Ans = Differences between deep learning and machine learning:

1. The main difference between deep learning and machine learning is due to the way data is presented in the system. Machine learning algorithms almost always require structured data, while deep learning networks rely on layers of ANN (artificial neural networks).
2. Machine learning algorithms are designed to “learn” to act by understanding labeled data and then use it to produce new results with more datasets. However, when the result is incorrect, there is a need to “teach them”.
3. Deep learning networks do not require human intervention, as multilevel layers in neural networks place data in a hierarchy of different concepts, which ultimately learn from their own mistakes. However, even they can be wrong if the data quality is not good enough.
4. Data decides everything. It is the quality of the data that ultimately determines the quality of the result.
5. Machine learning uses algorithms to parse data, learn from that data, and make informed decisions based on what it has learned.
6. Deep learning structures algorithms in layers to create an “artificial neural network” that can learn and make intelligent decisions on its own.
7. Deep learning is a subfield of machine learning. While both fall under the broad category of artificial intelligence, deep learning is what powers the most human-like artificial intelligence.

3. Define Precision and Recall.

Ans = Precision-Recall is a useful measure of success of prediction when the classes are very imbalanced. In information retrieval, **precision** is a measure of result relevancy, while **recall** is a measure of how many truly relevant results are returned.

4. When will you use Classification over Regression?

Ans = Classification is used when the output variable is a category such as “red” or “blue”, “spam” or “not spam”. It is used to draw a conclusion from observed values. Differently from, regression which is used when the output variable is a real or continuous value like “age”, “salary”, etc. When we must identify the class, the data belongs to we use classification over regression. Like when you must identify whether a name is male or female instead of finding out how they are correlated with the person.

5. What is Semi-supervised Machine Learning?

Ans = Semi-supervised machine learning is a combination of **supervised** and **unsupervised machine learning** methods. **Semi-supervised learning** is an approach to **machine learning** that combines a small amount of labeled data with a large amount of unlabeled data during training.

6. What is Decision Tree Classification?

Ans = Decision Tree - Classification. **Decision tree** builds **classification** or regression models in the form of a **tree** structure. It breaks down a dataset into smaller and smaller subsets while at the

same time an associated **decision tree** is incrementally developed. . **Decision trees** can handle both categorical and numerical data.

Some **uses of decision trees** are: Building knowledge management platforms for customer service that improve first call resolution, average handling time, and customer satisfaction rates. In finance, forecasting future outcomes and assigning probabilities to those outcomes.

7. What is pruning in Decision Trees, and how is it done?

Ans= Pruning is a technique in machine learning and search algorithms that reduces the size of **decision trees** by removing sections of the **tree** that provide little power to classify instances. **Pruning** reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.

Pre-pruning or early stopping involves stopping the tree before it has completed classifying the training set and **post-pruning** refers to **pruning** the tree after it has finished.

8. Briefly explain Logistic Regression.

Ans = Logistic regression is a statistical **model** that in its basic form uses a **logistic** function to **model** a binary dependent variable, although many more complex extensions exist.

In **regression** analysis, **logistic regression** (or **logit regression**) is estimating the parameters of a **logistic model** (a form of binary **regression**). Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical.

For example-

To predict whether an email is spam (1) or(0)

Whether the tumor is malignant (1) or not(0)

The **goal of logistic regression** is to correctly predict the category of outcome for individual cases using the most parsimonious model. To accomplish this **goal**, a model is created that includes all predictor variables that are useful in predicting the response variable.

9. What is a Recommendation System?

Ans = A recommender system, or a **recommendation system** (sometimes replacing 'system' with a synonym such as platform or **engine**), is a subclass of information filtering **system** that seeks to predict the "rating" or "preference" a user would give to an item. They are primarily used in commercial applications. Machine learning algorithms in recommender systems are typically classified into two categories— content based and collaborative filtering methods although modern recommenders combine both approaches.

10. What are some methods of reducing dimensionality?

Ans=Dimensionality reduction is simply, the process of reducing the dimension of your feature set. Your feature set could be a dataset with a hundred columns (i.e features) or it could be an array of points that make up a large sphere in the three-dimensional space. The most common and well known dimensionality reduction methods are the ones that apply linear transformations, like

1. PCA (Principal Component Analysis) : Popularly used for dimensionality reduction in continuous data, PCA rotates and projects data along the direction of increasing variance. The features with the maximum variance are the principal components.
2. Factor Analysis : a technique that is used to reduce a large number of variables into fewer numbers of factors. The values of observed data are expressed as functions of a number of possible causes in order to find which are the most important. The observations are assumed to be caused by a linear transformation of lower dimensional latent factors and added Gaussian noise.
3. LDA (Linear Discriminant Analysis): projects data in a way that the class separability is maximised. Examples from same class are put closely together by the projection. Examples from different classes are placed far apart by the projection

PART-B

1. What are the important objectives of machine learning? Discuss different important examples of machine Learning.

Ans = Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial **intelligence** based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.

The primary purpose of **machine learning** is to discover patterns in the user data and then make predictions based on these and intricate patterns for answering business questions and solving business problems. Machine learning helps in analysing the data as well as identifying trends.

The primary purpose of **machine learning** is to discover patterns in the user data and then make predictions based on these and intricate patterns for answering business questions and solving business problems. **Machine learning** helps in analysing the data as well as identifying trends. few examples of machine learning that we use everyday and perhaps have no idea that they are driven by ML.

1. Virtual Personal Assistants

Siri, Alexa, Google Now are some of the popular examples of virtual personal assistants. As the name suggests, they assist in finding information, when asked over voice. All you need to do is activate them and ask “What is my schedule for today?”, “What are the flights from Germany to London”, or similar questions. For answering, your personal assistant looks out for the information, recalls your related queries, or send a command to other resources (like phone apps) to collect info. You can even instruct assistants for certain tasks like “Set an alarm for 6 AM next morning”, “Remind me to visit Visa Office day after tomorrow”..

2. Predictions while Commuting

Traffic Predictions: We all have been using GPS navigation services. While we do that, our current locations and velocities are being saved at a central server for managing traffic. This data is then used to build a map of current traffic. While this helps in preventing the traffic and does congestion analysis, the underlying problem is that there are less number of cars that are equipped with GPS. Machine learning in such scenarios helps to estimate the regions where congestion can be found on the basis of daily experiences.

Online Transportation Networks: When booking a cab, the app estimates the price of the ride. When sharing these services, how do they minimize the detours? The answer is machine learning. Jeff

Schneider, the engineering lead at Uber ATC reveals in an interview that they use ML to define price surge hours by predicting the rider demand. In the entire cycle of the services, ML is playing a major role.

3. Videos Surveillance

Imagine a single person monitoring multiple video cameras! Certainly, a difficult job to do and boring as well. This is why the idea of training computers to do this job makes sense.

The video surveillance system nowadays are powered by AI that makes it possible to detect crime before they happen. They track unusual behaviour of people like standing motionless for a long time, stumbling, or napping on benches etc. The system can thus give an alert to human attendants, which can ultimately help to avoid mishaps. And when such activities are reported and counted to be true, they help to improve the surveillance services. This happens with machine learning doing its job at the backend.

4. Social Media Services

From personalizing your news feed to better ads targeting, social media platforms are utilizing machine learning for their own and user benefits. Here are a few examples that you must be noticing, using, and loving in your social media accounts, without realizing that these wonderful features are nothing but the applications of ML.

- **People You May Know:** Machine learning works on a simple concept: understanding with experiences. Facebook continuously notices the friends that you connect with, the profiles that you visit very often, your interests, workplace, or a group that you share with someone etc. On the basis of continuous learning, a list of Facebook users are suggested that you can become friends with.
- **Face Recognition:** You upload a picture of you with a friend and Facebook instantly recognizes that friend. Facebook checks the poses and projections in the picture, notices the unique features, and then matches them with the people in your friend list. The entire process at the backend is complicated and takes care of the precision factor but seems to be a simple application of ML at the front end.
- **Similar Pins:** Machine learning is the core element of Computer Vision, which is a technique to extract useful information from images and videos. Pinterest uses computer vision to identify the objects (or pins) in the images and recommend similar pins accordingly.

2. What are issues in decision tree learning? How are they overcome?

Ans= Issues in Decision Tree Learning

1. Overfitting the data:

$$h \in H$$

Definition: given a hypothesis space H , a hypothesis is said to **overfit** the training data if there

$$h' \in H$$

exists some alternative hypothesis, such that h has smaller error than h' over the training examples, but h' has smaller error than h over the entire distribution of instances.

2. Guarding against bad attribute choices:

The information gain measure has a bias that favors attributes with many values over those with only a few.

$$\forall v \in \text{Values}(\text{Date}) : H(S_v) = 0$$

For example, if you imagine an attribute *Date* with unique values for each training example, then $\text{Gain}(S, \text{Date})$ will yield $H(S)$ since

Obviously no other attribute can do better. This will result in a very broad tree of depth 1.

To guard against this, Quinlan uses $\text{GainRatio}(S, A)$ instead of $\text{Gain}(S, A)$.

$$\text{GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)}$$

where

$$\text{SplitInformation}(S, A) = - \sum_{v \in \text{Values}(A)} P(S_v) \log P(S_v)$$

with $P(S_v)$ estimated by relative frequency as before.

This introduces a penalty for partitioning into *equipossible* groups.

3. Handling continuous valued attributes:

Note that continuous valued attributes can be partitioned into a discrete number of disjoint intervals. Then we can test for membership to these intervals.

If the *Temperature* attribute in the *PlayTennis* example took on continuous values in the range 40-90, note that we cannot just use the same approach as before.

Why? Because *Temperature* becomes a bad choice for classification (It alone may perfectly classify the training examples and therefore promise the highest information gain like in the earlier *Date* example) while remaining a poor predictor on the test set.

The solution to this problem is to classify based not on the actual temperature, but on dynamically determined intervals within which the temperature falls.

$$T \leq a \quad a < T \leq b \quad b < T \leq c$$

For instance we can introduce boolean attributes $T \leq a$, $a < T \leq b$, $b < T \leq c$,

and $T > c$ instead of real valued T . a , b and c can be computed dynamically based on boundaries where T is likely or known to change.

4. Handling missing attribute values:

What happens if some of the training examples contain one or more "?", meaning "value not known" instead of the actual attribute values?

Here are some common *ad hoc* solutions:

- Substitute "?" by the most common value in that column.
- Substitute "?" by the most common value among all training examples that have been sorted into the tree at that node.

- Substitute "?" by the most common value among all training examples that have been sorted into the tree at that node with the same classification as the incomplete example.

Quinlan's solution in C4.5 is slightly more complex: Instead of a value, he assigns a distribution over values to the "?". This distribution is used when computing $Gain(S,A)$. The distribution determines how much of $H(S_v=?)$ will contribute to each $H(S_v)$.

5. Handling attributes with differing costs:

Sometimes we want to introduce additional bias against the selection of certain attributes. Eg. "Wherever possible try not to use *InvasiveTest* as an attribute, since this might inconvenience the patient.

Here is another *ad hoc* heuristic to fix this:

Introduce a user-defined measure $Cost(A)$, to specify a fixed bias against each attribute.

$$CostedGain(S, A) = \frac{Gain^2(S, A)}{Cost(A)} \quad \text{or}$$

$$CostedGain(S, A) = \frac{2^{Gain(S,A)} - 1}{(Cost(A) + 1)^w}$$

Now we can use a $CostedGain(S,A)$ which is defined along the lines of:

3. What are the steps in Back propagation algorithm? Why a Multilayer neural network is required?

Ans= The **Backpropagation algorithm** is a supervised learning method for multilayer feed-forward networks from the field of Artificial Neural Networks. Feed-forward neural networks are inspired by the information processing of one or more neural cells, called a neuron.

The application algorithm for BPN is shown below:

STEP 1: Initialize weights (from training algorithm).

STEP 2: For each input vector do steps 3-5.

STEP 3: For $i=1, \dots, n$; set activation of input unit, X_i ;

STEP 4: For $j=1, \dots, p$;
 $Z_{inj} = V_{oj} + \sum_i X_i V_{ij}$

STEP 5: For $k=1, \dots, m$
 $Y_{ink} = W_{ok} + \sum_j Z_j W_{jk}$
 $Y_k = f(Y_{ink})$

Multilayer networks solve the classification problem for non linear sets by employing hidden layers, whose neurons are not directly connected to the output. The additional hidden layers can be interpreted geometrically as additional hyper-planes, which enhance the separation capacity of the **network**.

Since there are **multiple layers** of neurons, **MLP** is a deep learning technique. **MLP** is widely used for solving problems that **require** supervised learning as well as research into computational neuroscience and parallel distributed processing.

4. Describe k-nearest neighbor algorithm. Why is it called instance based learning?

Ans= The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

The KNN Algorithm

1. Load the data
2. Initialize K to your chosen number of neighbors
3. 3. For each example in the data
 - 3.1 Calculate the distance between the query example and the current example from the data.
 - 3.2 Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sortedcollection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

The raw training **instances** are used to make predictions. As such **KNN** is often referred to as **instance- based learning** or a **case-based learning** (where each training **instance** is a case from the problem domain). The raw training **instances** are used to make predictions. As such **KNN** is often referred to as **instance-based learning** or a **case-based learning** (where each training **instance** is a case from the problem domain).

5. Describe these terms in brief (I) PAC Hypothesis (II) Mistake bound model of learning

Ans- (I) PAC Hypothesis: Probably approximately correct (**PAC**) **learning** is a theoretical framework for analyzing the generalization error of a **learning** algorithm in terms of its error on a training set and some measure of complexity. The goal is typically to show that an algorithm achieves low generalization error with high probability.

The intent of the PAC model is that successful learning of an unknown target concept should entail obtaining, with high probability, a hypothesis that is a good approximation of it. Hence the name Probably Approximately Correct. In the basic model, the instance space is assumed to be $\{0,1\}^n$, the set of all possible assignments to n Boolean variables (or *attributes*) and concepts and hypotheses are subsets of $\{0,1\}^n$. The notion of approximation is defined by assuming that there is some probability distribution D defined on the instance space $\{0,1\}^n$, giving the probability of each instance. We then let the *error* of a hypothesis h w.r.t. a fixed target concept c , denoted $error(h)$ when c is clear from the context, be defined by

$$error(h) = \sum_{x \in h \Delta c} D(x),$$

where Δ denotes the symmetric difference. Thus, $error(h)$ is the probability that h and c will disagree on an instance drawn randomly according to D . The hypothesis h is a good approximation

(II) Mistake bound model of learning

In the MB model, learning is in stages. In each stage:

1. The learner gets an unlabeled example.
2. The learner predicts its classification.
3. The learner is told the correct label.

Goal: bound the total number of mistakes.

DEFINITION: Algorithm A has mistake-bound M for learning class C if A makes at most M mistakes on any sequence that is consistent with a function in C .

Notice that since we're not assuming anything about how examples are selected, we can't necessarily talk about "how much data do I need to converge". E.g., maybe we end up seeing the same example over and over again and don't learn anything new. But, that's OK because we (hopefully) won't be making mistakes either. Later on, when we talk about distributional models, we will be able to convert mistake bounds into bounds on how much data we need to converge. But the mistake-bound setting is nice because it's very clean -- no probabilities. So this is the toughest model for positive results.

We'll think of A as being good if its mistake bound is polynomial in n = the size of the examples, and s = the description length of the smallest consistent function (for classes like DNF and Decision Trees that can have very complicated functions in them). [We are thinking of a "concept class" as both a set of functions and a description language for them.]

DEFINITION: C is **learnable** in the mistake bound model if there exists an algorithm with mistake bound $\text{poly}(n,s)$, and furthermore whose running time per stage is $\text{poly}(n,s)$.

6. What do you mean by Gain and Entropy? How is it used to build the Decision tree in algorithm? Illustrate using an example

Ans= Information Gain: When we use a node in a decision tree to partition the training instances into smaller subsets the entropy changes. Information gain is a measure of this change in entropy.

Definition: Suppose S is a set of instances, A is an attribute, S_v is the subset of S with $A = v$, and Values (A) is the set of all possible values of A, then Entropy,

Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content.

Definition: Suppose S is a set of instances, A is an attribute, S_v is the subset of S with $A = v$, and Values (A) is the set of all possible values of A, then Example:

For the set $X = \{a,a,a,b,b,b,b\}$

Total instances: 8

Instances of b: 5

Instances of a: 3

$$= -[0.375 * (-1.415) + 0.625 * (-0.678)]$$

$$= -(-0.53 - 0.424)$$

$$= 0.954$$

Building Decision Tree using Information

Gain The essentials:

Start with all training instances associated with the root node

Use info gain to choose which attribute to label each node with

Note: No root-to-leaf path should contain the same discrete attribute twice

Recursively construct each subtree on the subset of training instances that

- would be classified down that path in the tree.

The border cases:

- If all positive or all negative training instances remain, label that node “yes” or “no” accordingly
- If no attributes remain, label with a majority vote of training instances left at that node
- If no instances remain, label with a majority vote of the parent’s training instances

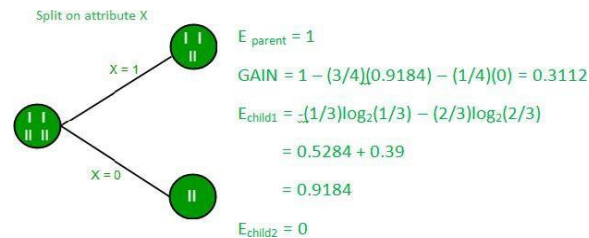
Example:

Now, lets draw a Decision Tree for the following data using Information gain.

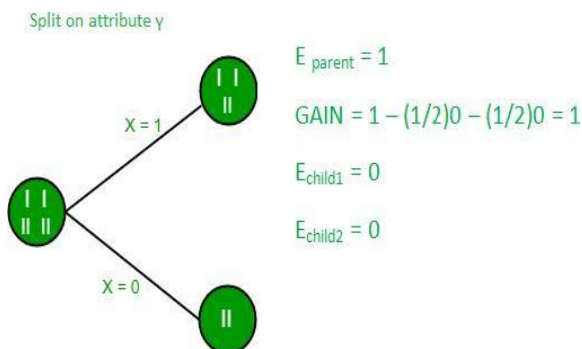
- **Training set: 3 features and 2 classes**

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

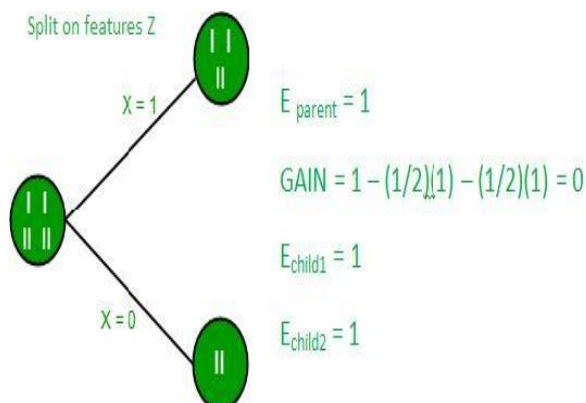
- Here, we have 3 features and 2 output classes.
To build a decision tree using Information gain. We will take each of the feature and calculate the information for each feature.



- Split on feature X**

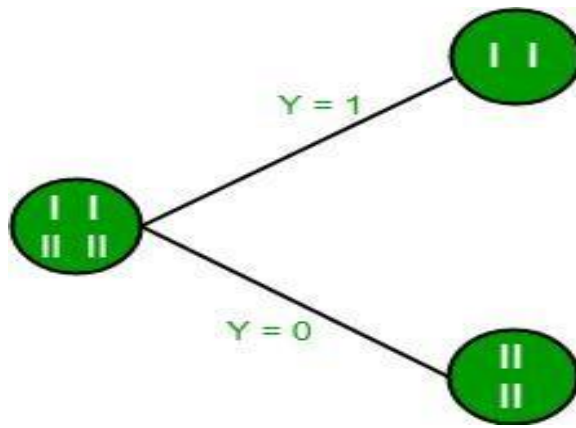


- Split on feature Y**



- Split on feature Z**

- From the above images we can see that the information gain is maximum when we make a split on feature Y. So, for the root node best suited feature is feature Y. Now we can see that while splitting the dataset by feature Y, the child contains pure subset of the target variable. So we don't need to further split the dataset.
- The final tree for the above dataset would be look like this:



7. Describe in brief: I) Cross-over operators and mutations. II) Case- based reasoning

Ans = II) Case- based reasoning

CBR is reasoning by remembering: previously solved problems (cases) are used to suggest solutions for novel but similar problems. Kolodner [6] lists four assumptions about the world around us that represent the basis of the CBR approach:

1. Regularity: the same actions executed under the same conditions will tend to have the same or similar outcomes.
2. Typicality: experiences tend to repeat themselves.
3. Consistency: small changes in the situation require merely small changes in the interpretation and in the solution.
4. Adaptability: when things repeat, the differences tend to be small, and the small differences are easy to compensate for.

In the problem solving illustrated in Fig. 1 and explained above, the following steps were taken: describing the current problem, searching for a similar previously solved problem, retrieving the solution to it, adapting the solution to the current problem, verifying the solution, and storing the newly solved problem. In turn, since the newly found solution may be used for solving future problems, the process illustrated in Fig. 1 denotes, in fact, the CBR working cycle. According to Kolodner, the CBR working cycle can be described best in terms of four processing stages:

1. Case retrieval: after the problem situation has been assessed, the best matching case is searched in the case base and an approximate solution is retrieved.
2. Case adaptation: the retrieved solution is adapted to fit better the new problem.
3. Solution evaluation: the adapted solution can be evaluated either before the solution is applied to the problem or after the solution has been applied. In any case, if the accomplished result is not satisfactory, the retrieved solution must be adapted again or more cases should be retrieved.
4. Case-base updating: If the solution was verified as correct, the new case may be added to the case base.

: I) Cross-over operators and mutations

Crossover and mutation perform two different roles. Crossover (like selection) is a convergence operation which is intended to pull the population towards a local minimum/maximum. In an

interesting aside, crossover is not one of the original genetic operators; Holland's original thesis used only selection and probabilistic mutation.

Mutation is a divergence operation. It is intended to occasionally break one or more members of a population out of a local minimum/maximum space and potentially discover a better minimum/maximum space.

Since the end goal is to bring the population to convergence, selection/crossover happen more frequently (typically every generation). Mutation, being a divergence operation, should happen less frequently, and typically only effects a few members of a population (if any) in any given generation.

PART- C

1. Discuss linear regression in detail with an example.

Ans= **Linear regression** quantifies the relationship between one or more predictor variable(s) and one outcome variable. Linear regression is commonly used for predictive analysis and modeling. For example, it can be used to quantify the relative impacts of age,gender, and diet (the predictor variables) on height (the outcome variable). Linear regression is also known as multiple regression, multivariate regression, ordinary least squares (OLS),and regression. This post will show you examples of linear regression, including an example of simple linear regression and an example of multiple linear regression.

Example of simple linear regression

The table below shows some data from the early days of the Italian clothing company Benetton. Each row in the table shows Benetton's sales for a year and the amount spent on advertising that year. In this case, our outcome of interest is sales—it is what we want to predict. If we use advertising as the predictor variable, linear regression estimates that **Sales = 168 + 23 Advertising**. That is, if advertising expenditure is increased by one Euro, then sales will be expected to increase by 23 million Euros, and if

Year	Sales (Million Euro)	Advertising (Million Euro)
1	651	23
2	762	26
3	856	30
4	1,063	34
5	1,190	43
6	1,298	48
7	1,421	52
8	1,440	57
9	1,518	58

there was no advertising we would expect sales of 168 million Euros.

Example of multiple linear regression

Linear regression with a single predictor variable is known as simple regression. In real-world applications, there is typically more than one predictor variable. Such regressions are called multiple regression. For more information, check out this post on why you should not use multiple linear regression for Key Driver Analysis with example data for multiple linear regression examples.

Returning to the Benetton example, we can include **year** variable in the regression, which gives the result that **Sales = 323 + 14 Advertising + 47 Year**. The interpretation of this equation is that every extra million Euro of advertising expenditure will lead to an extra 14 million Euro of sales and that sales will grow due to non-advertising factors by 47 million Euro per year.

2. (a) Which is more important model accuracy or model performance? Support with suitable example.

Ans= In analysis of medical images to determine if there is a disease (such as cancer), the accuracy extremely critical, even if the models would take minutes or hours to make a prediction.

Other applications require real time performance, even if this comes at a cost of accuracy. For example, imagine a machine that views a fast conveyor belt carrying tomatoes, where it must separate the green from the red ones. Though an occasional error is undesired, the success of this machine is more determined by its ability to withstand its throughput.

A more common example is face detection for recreational applications. People would expect a fast response from the app, though the occasional missed face would not render it useless.

For example, assume that a person makes a withdrawal of Rs. 10000 from an ATM machine, and the machines dispatches amount worth Rs. 9000. The system cannot be said to be not performing, as it is producing a response by dispatching the money. However, the accuracy of the system is not satisfactory because the system did not produce correct response.

Specifically talking on machine learning models, in the current real-world scenarios, the systems/models with highest accuracies survive and are used globally. Models which simply perform and produce reasonable/medium accuracies are termed OTHERS.

(b) Explain k-fold cross validation does not work well with time series model. How it can be resolved.

Ans= Time-series (or other intrinsically ordered data) can be problematic for cross-validation. If some pattern emerges in year 3 and stays for years 4-6, then your model can pick up on it, even though it wasn't part of years 1 & 2.

An approach that's sometimes more principled for time series is forward chaining, where your procedure would be something like this:

fold 1 : training [1], test [2]

fold 2 : training [1 2], test [3]

fold 3 : training [1 2 3], test [4]

fold 4 : training [1 2 3 4], test [5]

fold 5 : training [1 2 3 4 5], test [6]

That more accurately models the situation you'll see at prediction time, where you'll model on past data and predict on forward-looking data. It also will give you a sense of the dependence of your modeling on data size.

The "canonical" way to do time-series cross-validation (at least as described by @Rob Hyndman) is to "roll" through the dataset.

i.e.:

fold 1 : training [1], test [2]

fold 2 : training [1 2], test [3]

fold 3 : training [1 2 3], test [4]

fold 4 : training [1 2 3 4], test [5]

fold 5 : training [1 2 3 4 5], test [6]

Basically, your training set should not contain information that occurs after the test set.

3. Explain the Q function and Q Learning Algorithm assuming deterministic rewards and actions with example.

Ans= **Q-Learning** is an example of model-free learning algorithm. It does not assume that agent knows anything about the state-transition and reward models. However, the agent will discover what are the good and bad actions by trial and error.

The basic idea of Q-Learning is to approximate the state-action pairs Q-function from the samples of $Q(s, a)$ that we observe during interaction with the environment. This approach is known as **Time-Difference Learning**.

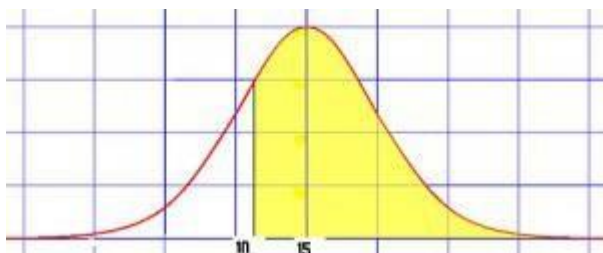
$$Q(s, a) = (1 - \alpha) Q(s, a) + \alpha Q_{obs}(s, a)$$

where, $Q_{obs}(s, a) = r(s, a) + \gamma \max_{a'} Q(s', a')$

Q-Learning algorithm

where α is the learning rate. The $Q(s,a)$ table is initialized randomly. Then the agent starts to interact with the environment, and upon each interaction the agent will observe the reward of its action $r(s,a)$ and the state transition (new state s'). The agent compute the observed Q-value $Q_{obs}(s, a)$ and then use the above equation to update its own estimate of $Q(s,a)$.

The Q function is just one minus the cumulative distribution function (CDF) for the standardized normal distribution. In other words, it gives you the right tail area.



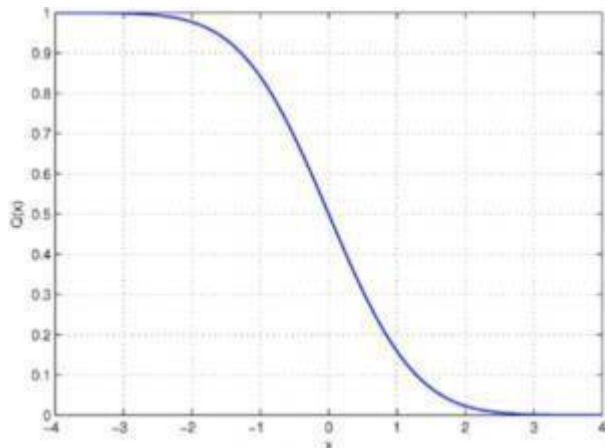
The area of a right tail in the normal distribution (yellow area) can be found with the Q function.

The CDF for the normal distribution gives you the probability that a normal random variable takes a value equal to or smaller than x . The Q function is the complement of this; In other words, it's the probability a normal random variable takes a value greater than x .

As a formula:

$$Q(x) = 1 - \text{CDF} = P(X > x)$$

The plot starts with an area of 1, representing 100% probability. At the point on the far left of the bell



curve, the right “tail” is actually the entire area of the curve.

Plot of the Q Function. The y-axis represents probabilities from 0 to 1. The x-axis represents standard deviations, or z-scores.

Calculating the function by hand is relatively simple: find the CDF, and subtract from one. Some software programs find the Q function directly. For example, In **MATLAB**, the syntax is $y = \text{qfunc}(x)$. If your software doesn't, find the CDF and subtract from one.

4. (a) Explain Naïve bayes classifier with an example

Ans= A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

Bayes Theorem:

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

Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

Example:

Let us take an example to get some better intuition. Consider the problem of playing golf. The dataset is represented as below.

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes

We classify whether the day is suitable for playing golf, given the features of the day. The columns represent these features and the rows represent individual entries. If we take the first row of the dataset, we can observe that is not suitable for playing golf if the outlook is rainy, temperature is hot, humidity is high and it is not windy. We make two assumptions here, one as stated above we consider that these predictors are independent. That is, if the temperature is hot, it does not necessarily mean that the humidity is high. Another assumption made here is that all the predictors have an equal effect on the outcome. That is, the day being windy does not have more importance in deciding to play golf or not.

According to this example, Bayes theorem can be rewritten as:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

The variable **y** is the class variable(play golf), which represents if it is suitable to play golf or not given the conditions. Variable **X** represent the parameters/features.

X is given as,

$$X = (x_1, x_2, x_3, \dots, x_n)$$

Here x_1, x_2, \dots, x_n represent the features, i.e they can be mapped to outlook, temperature, humidity and windy. By substituting for **X** and expanding using the chain rule we get,

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y)\dots P(x_n|y)P(y)}{P(x_1)P(x_2)\dots P(x_n)}$$

Now, you can obtain the values for each by looking at the dataset and substitute them into the equation. For all entries in the dataset, the denominator does not change, it remain static.

Therefore, the denominator can be removed and a proportionality can be introduced.

$$P(y|x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i|y)$$

In our case, the class variable(**y**) has only two outcomes, yes or no. There could be cases where the classification could be multivariate. Therefore, we need to find the class **y** with maximum probability.

$$y = \operatorname{argmax}_y P(y) \prod_{i=1}^n P(x_i|y)$$

Using the above function, we can obtain the class, given the predictors.

(b) Define the following terms

i) Sample error

ans= The sample error, denoted $errors(h)$ of hypothesis h with respect to target function f and data sample S is :

$$errors(h) = \frac{1}{n} \sum_{x \in S} \delta(f(x), h(x)) \quad errors(h) = \frac{1}{n} \sum_{x \in S} \delta(f(x), h(x))$$

where n is the number of examples in S , and the quantity

$$\delta(f(x), h(x)) = \begin{cases} 1 & \text{if } f(x) \neq h(x) \\ 0 & \text{otherwise} \end{cases}$$

ii) True error

ans= The **true error**, denoted $error_D(h)$ of hypothesis with respect to target function f and distribution D , is the probability that h will misclassify an instance drawn a random according to D :
 $error_D(h) = P(f(x) \neq h(x))$

The true error, denoted $error_D(h)$ of hypothesis with respect to target function f and distribution D , is the probability that h will misclassify an instance drawn a random according to D :

$$error_D(h) = P(f(x) \neq h(x)) \quad error_D(h) = P(f(x) \neq h(x))$$

iii) Expected value.

Ans= Expected value refers to an interesting techniques that is often used in **machine learning** theory, statistics, probability analysis or during general big data analysis. The idea is to use a probability in order to tell what outcomes can be **expected** in the long run.

5. What is Artificial Neural Network? Explain appropriate problem for Neural Network learning with its characteristics.

Ans= Artificial neural networks (ANN) or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains.^[1] Such systems "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge of cats, for example, that they have fur, tails, whiskers and cat-like faces. Instead, they automatically generate identifying characteristics from the examples that they process.

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it.

Overfitting/underfitting: **Neural networks** normally have a large enough capacity to store the whole training data, including noise. There are many ways like dropout, & regularization to work around such a **problem** but still, modern **neural networks** have a tendency to overfit.

- Training a neural network involves using an optimization algorithm to find a set of weights to best map inputs to outputs.
- The problem is hard, not least because the error surface is non-convex and contains local minima, flat spots, and is highly multidimensional.
- The stochastic gradient descent algorithm is the best general algorithm to address this challenging problem.

Model Paper 5

PART-A

Ques 1. Explain Linear Regression with example.

Ans. Linear regression quantifies the relationship between one or more predictor variable(s) and one outcome variable. Linear regression is commonly used for predictive analysis and modeling. For example, it can be used to quantify the relative impacts of age, gender, and diet (the predictor variables) on height (the outcome variable). Linear regression is also known as multiple regression, multivariate regression, ordinary least squares (OLS), and regression.

Ques 2. Differentiate between supervised and unsupervised learning.

Ans. Difference b/w Supervised and Unsupervised Learning :

	SUPERVISED LEARNING	UNSUPERVISED LEARNING
Input Data	Uses Known and Labeled Data as input	Uses Unknown Data as input
Computational Complexity	Very Complex	Less Computational Complexity
Real Time	Uses off-line analysis	Uses Real Time Analysis of Data
Number of Classes	Number of Classes are known	Number of Classes are not known
Accuracy of Results	Accurate and Reliable Results	Moderate Accurate and Reliable Results

Ques 3. Define perceptron.

Ans. A Perceptron is an algorithm used for supervised learning of binary classifiers. Binary classifiers decide whether an input, usually represented by a series of vectors, belongs to a specific class.

In short, a perceptron is a single-layer neural network. They consist of four main parts including input values, weights and bias, net sum, and an activation function.

Ques 4. What is dimensionality reduction?

Ans. Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction.

An intuitive example of dimensionality reduction can be discussed through a simple e-mail classification problem, where we need to classify whether the e-mail is spam or not. This can involve a large number of features, such as whether or not the e-mail has a generic title, the content of the e-mail, whether the e-mail uses a template, etc.

Ques 5. Distinguish between classification and regression?

Classification vs Regression

- | | |
|---|---|
| <ul style="list-style-type: none">• Classification means to group the output into a class.• classification to predict the type of tumor i.e. harmful or not harmful using training data• if it is discrete/categorical variable, then it is classification problem | <ul style="list-style-type: none">• Regression means to predict the output value using training data.• regression to predict the house price from training data• if it is a real number/continuous, then it is regression problem. |
|---|---|

Ques 6. What is the principle of Markov decision method?

Ans. The Markov property states that, “The future is independent of the past given the present.” Once the current state is known, the history of information encountered so far may be thrown away, and that state is a sufficient statistic that gives us the same characterization of the future as if we have all the history.

Ques 7. In which algorithm Gini index is used. Explain that algorithm in brief.

Ans. ID3 algorithm uses information gain for constructing the decision tree. Gini Index: It is calculated by subtracting the sum of squared probabilities of each class from one. It favors larger partitions and is easy to implement whereas information gain favors smaller partitions with distinct values.

Ques 8. Define random forest.

Ans. The random forest is a supervised learning algorithm that randomly creates and merges multiple decision trees into one “forest.” The goal is not to rely on a single learning model, but rather a collection of

decision models to improve accuracy. The primary difference between this approach and the standard decision tree algorithms is that the root nodes feature splitting nodes are generated randomly.

Ques 9. Define Precision and Recall.

Ans. In pattern recognition, information retrieval and classification (machine learning), precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of the total amount of relevant instances that were actually retrieved. Both precision and recall are therefore based on an understanding and measure of relevance.

Ques 10. What is Overfitting and how can you avoid it?

Ans. Overfitting :- A statistical model is said to be overfitted, when we train it with a lot of data. When a model gets trained with so much of data, it starts learning from the noise and inaccurate data entries in our data set. Then the model does not categorize the data correctly, because of too much of details and noise.

How to avoid Overfitting:

The commonly used methodologies are:

- Cross- Validation:
- Early Stopping:
- Pruning:
- Regularization:

PART- B

Ques 1. (a) What is a training set and how is it used to train neural networks?

Ans. Training set is a set of pairs of input patterns with corresponding desired output patterns. Each pair represents how the network is supposed to respond to a particular input. The network is trained to respond correctly to each input pattern from the training set. Training algorithms that use training sets are called supervised learning algorithms. We may think of a supervised learning as learning with a teacher, and the training set as a set of examples. During training the network, when presented with input patterns, gives 'wrong' answers (not desired output). The error is used to adjust the weights in the network so that next time the error was smaller. This procedure is repeated using many examples (pairs of inputs and desired outputs) from the training set until the error becomes sufficiently small.

Ques 1. (b) Describe the main steps of the supervised training algorithm?

Ans. In order to solve a given problem of supervised learning, one has to perform the following steps:

Determine the type of training examples. Before doing anything else, the user should decide what kind of data is to be used as a training set. In the case of handwriting analysis, for example, this might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.

Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.

Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.

Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support vector machines or decision trees.

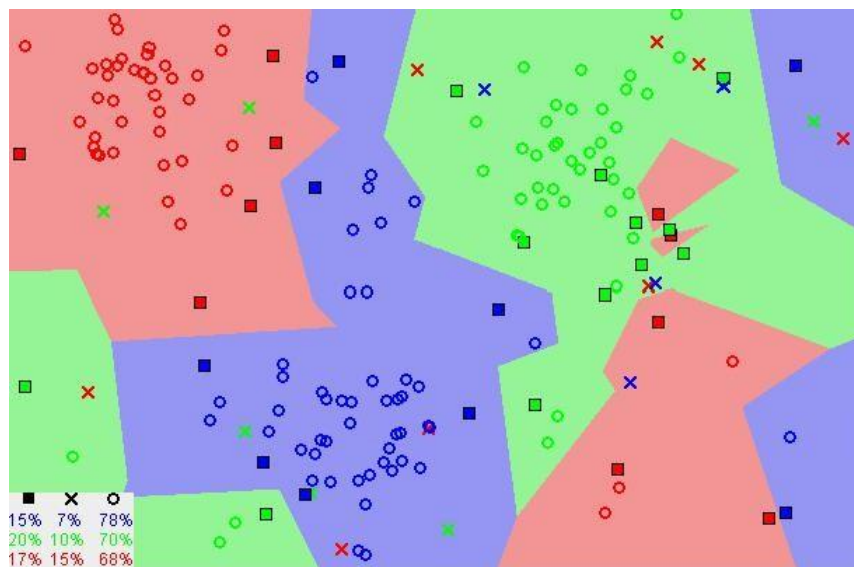
Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a validation set) of the training set, or via cross-validation.

Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set. Training data already trained.

Ques 2. Enumerate the steps in K-nearest neighbour algorithm.

Ans. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

“Birds of a feather flock together.”



Notice in the image above that most of the time, similar data points are close to each other. The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph.

There are other ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, the straight-line distance (also called the Euclidean distance) is a popular and familiar choice.

The KNN Algorithm

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data
 - 3.1 Calculate the distance between the query example and the current example from the data.
 - 3.2 Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

Advantages

1. The algorithm is simple and easy to implement.
2. There's no need to build a model, tune several parameters, or make additional assumptions.
3. The algorithm is versatile. It can be used for classification, regression, and search (as we will see in the next section).

Disadvantages

The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

Ques 3. Explain policy iteration and value iteration in brief.

Ans. Value Iteration

Value iteration computes the optimal state value function by iteratively improving the estimate of $V(s)$. The algorithm initialize $V(s)$ to arbitrary random values. It repeatedly updates the $Q(s, a)$ and $V(s)$ values until they converges. Value iteration is guaranteed to converge to the optimal values. This algorithm is shown in the following pseudo-code:

```

Initialize  $V(s)$  to arbitrary values
Repeat
  For all  $s \in S$ 
    For all  $a \in \mathcal{A}$ 
       $Q(s, a) \leftarrow E[r|s, a] + \gamma \sum_{s' \in S} P(s'|s, a)V(s')$ 
     $V(s) \leftarrow \max_a Q(s, a)$ 
Until  $V(s)$  converge

```

Policy Iteration

While value-iteration algorithm keeps improving the value function at each iteration until the value-function converges. Since the agent only cares about the finding the optimal policy, sometimes the optimal policy will converge before the value function. Therefore, another algorithm called policy-iteration instead of repeated improving the value-function estimate, it will re-define the policy at each step and compute the value according to this new policy until the policy converges. Policy iteration is also guaranteed to converge to the optimal policy and it often takes less iterations to converge than the value-iteration algorithm.

```

Initialize a policy  $\pi'$  arbitrarily
Repeat
   $\pi \leftarrow \pi'$ 
  Compute the values using  $\pi$  by
    solving the linear equations
     $V^\pi(s) = E[r|s, \pi(s)] + \gamma \sum_{s' \in S} P(s'|s, \pi(s))V^\pi(s')$ 
  Improve the policy at each state
     $\pi'(s) \leftarrow \arg \max_a (E[r|s, a] + \gamma \sum_{s' \in S} P(s'|s, a)V^\pi(s'))$ 
Until  $\pi = \pi'$ 

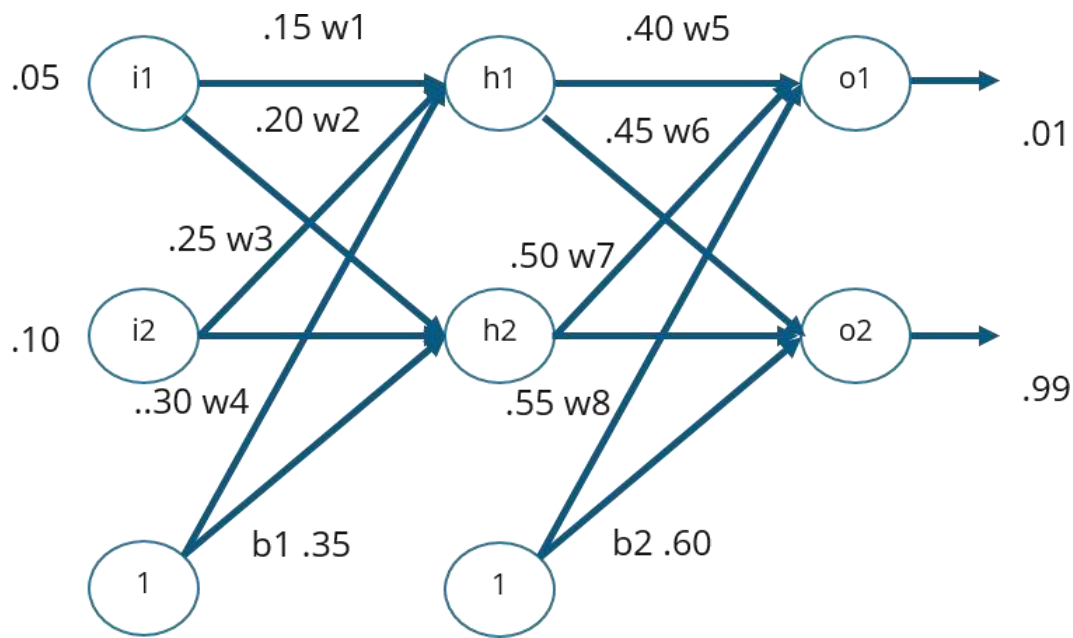
```

Ques 4. What are the steps in Back propagation algorithm? Why a Multilayer neural network is required?

Ans. Back propagation : The Backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function is then considered to be a solution to the learning problem.

How Backpropagation Works?

Consider the below Neural Network:



The above network contains the following:

- two inputs
- two hidden neurons
- two output neurons
- two biases

Below are the steps involved in Backpropagation:

- Step – 1: Forward Propagation
- Step – 2: Backward Propagation
- Step – 3: Putting all the values together and calculating the updated weight value

Backpropagation Algorithm:

```

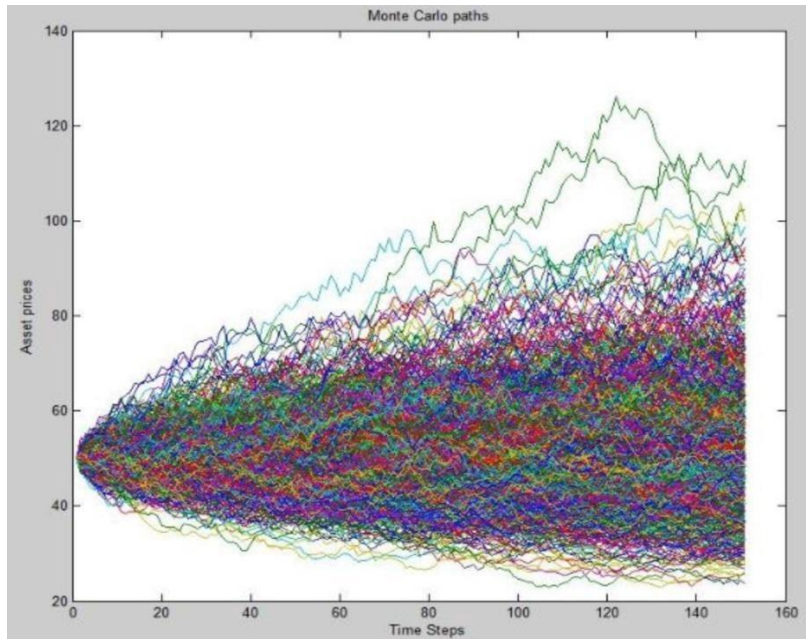
initialize network weights (often small random values)
do
  forEach training example named ex
    prediction = neural-net-output(network, ex) // forward pass
    actual = teacher-output(ex)
    compute error (prediction - actual) at the output units
    compute {displaystyle Delta w_{h}} for all weights from hidden layer to output layer //
backward pass
  compute {displaystyle Delta w_{i}} for all weights from input layer to hidden layer // backward
pass continued
  update network weights // input layer not modified by error estimate
until all examples classified correctly or another stopping criterion satisfied
return the network

```

Ques 5. Discuss Monte Carlo method in detail.

Ans. Monte Carlo (MC) methods are a subset of computational algorithms that use the process of repeated random sampling to make numerical estimations of unknown parameters. They allow for the modeling of

complex situations where many random variables are involved, and assessing the impact of risk. The uses of MC are incredibly wide-ranging, and have led to a number of groundbreaking discoveries in the fields of physics, game theory, and finance. There are a broad spectrum of Monte Carlo methods, but they all share the commonality that they rely on random number generation to solve deterministic problems. I hope to outline some of the basic principles of MC, and perhaps infect you with a bit of the excitement that I have about their possible applications.



The concept was invented by Stanislaw Ulam, a mathematician who devised these methods as part of his contribution to the Manhattan Project. He used the tools of random sampling and inferential statistics to model likelihoods of outcomes, originally applied to a card game (Monte Carlo Solitaire). Ulam later worked with collaborator John von Neumann, using newly developed computer technologies to run simulations to better understand the risks associated with the nuclear project. As you can imagine, modern computational technology allows us to model much more complex systems, with a larger number of random parameters, like so many of the scenarios that we encounter during our everyday lives. Before we consider the complicated systems however, let's talk about a simple case; a game of blackjack.

If we wanted to find the probability of getting blackjack (an ace along with a ten-valued card), we could simply count the number of possible hands where this is the case, and divide by the total number of possible combinations of cards to get the probability (it's around $1/21$, if you are curious). But now imagine our sample space is much harder to compute, for example our deck of cards has thousands as opposed to just 52 cards, or better yet we don't even know how many cards there are. There is another way to find this probability.

Ques 6. Describe k-nearest neighbor algorithm. Why is it called instance based learning?

Ans. INSTANCE-BASE LEARNING :- Instance-based learning methods simply store the training examples instead of learning explicit description of the target function. – Generalizing the examples is postponed until a new instance must be classified. – When a new instance is encountered, its relationship to the stored examples is examined in order to assign a target function value for the new instance.

- Instance-based learning includes nearest neighbor, locally weighted regression and case-based reasoning methods.
- Instance-based methods are sometimes referred to as lazy learning methods because they delay processing until a new instance must be classified.
- A key advantage of lazy learning is that instead of estimating the target function once for the entire instance space, these methods can estimate it locally and differently for each new instance to be classified.

k-Nearest Neighbor Learning :-

- k-Nearest Neighbor Learning algorithm assumes all instances correspond to points in the n-dimensional space R^n
- The nearest neighbors of an instance are defined in terms of Euclidean distance.

Euclidean distance between the instances x_i and x_j are:

- For a given query instance x_q , $f(x_q)$ is calculated the function values of k-nearest neighbor of $x_q = \sum_{i=1}^k f(x_i)$

Store all training examples

- Calculate $f(x_q)$ for a given query instance x_q using k-nearest neighbor
- Nearest neighbor: (k=1)
 - Locate the nearest training example x_n , and estimate $f(x_q)$ as
 - $f(x_n) \leftarrow f(x_q)$

k-Nearest neighbor:

- Locate k nearest training examples, and estimate $f(x_q)$ as
- If the target function is real-valued, take mean of f-values of k nearest neighbors.
- If the target function is discrete-valued, take a vote among f-values of k nearest neighbors.

When To Consider Nearest Neighbor

- Instances map to points in R^n
- Less than 20 attributes per instance
- Lots of training data
- Advantages
 - Training is very fast
 - Learn complex target functions

- Can handle noisy data
- Does not lose any information
- Disadvantages
 - Slow at query time
 - Easily fooled by irrelevant attributes

Ques 7. What do you mean by Gain and Entropy? How is it used to build the Decision tree in algorithm? Illustrate using an example.

Ans. Definition: Entropy is the measures of **impurity, disorder** or **uncertainty** in a bunch of examples. Entropy controls how a Decision Tree decides to **split** the data. It actually effects how a **Decision Tree** draws its boundaries.

The Equation of Entropy:

$$Entropy = - \sum p(X) \log p(X)$$



here $p(x)$ is a fraction of
examples in a given class

Definition: Information gain (IG) measures how much “information” a feature gives us about the class.

Why it matter ?

- **Information gain** is the main key that is used by **Decision Tree Algorithms** to construct a Decision Tree.
- **Decision Trees** algorithm will always tries to maximize **Information gain**.
- An **attribute** with highest **Information gain** will tested/split first.

The Equation of Information gain:

$$\text{Information gain} = \text{entropy (parent)} - [\text{weightes average}] * \text{entropy (children)}$$

Building Decision Tree using Information Gain

The essentials:

- Start with all training instances associated with the root node
- Use info gain to choose which attribute to label each node with
- Note: No root-to-leaf path should contain the same discrete attribute twice
- Recursively construct each subtree on the subset of training instances that would be classified down that path in the tree.

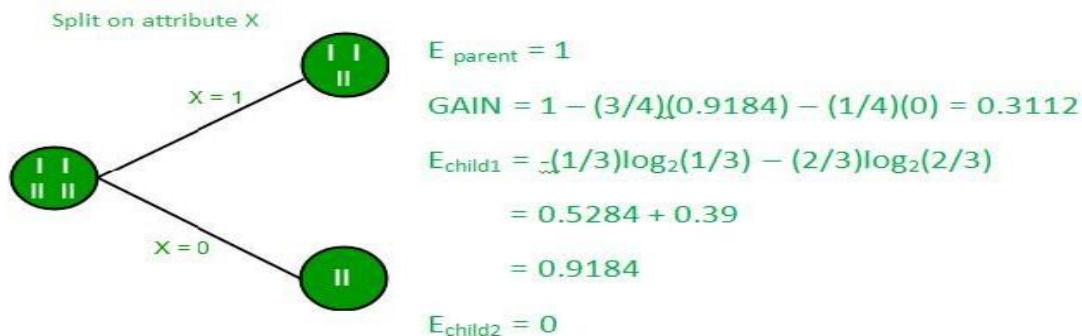
Example:

Training set: 3 features and 2 classes

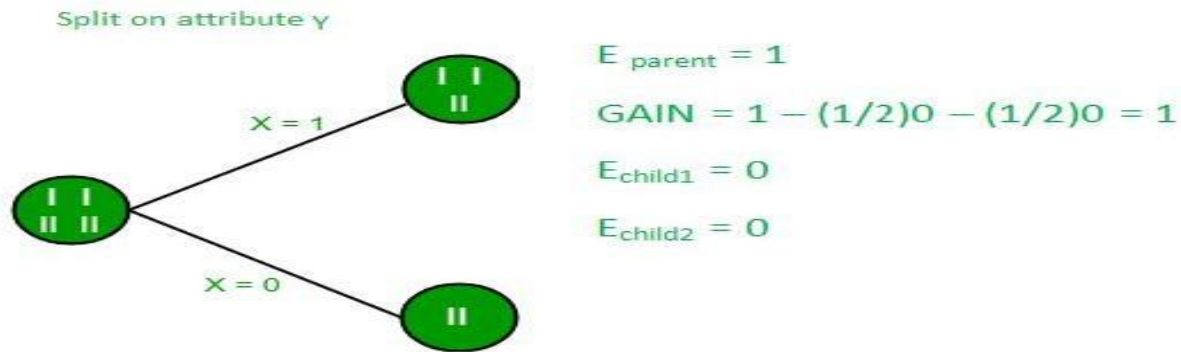
X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Here, we have 3 features and 2 output classes.

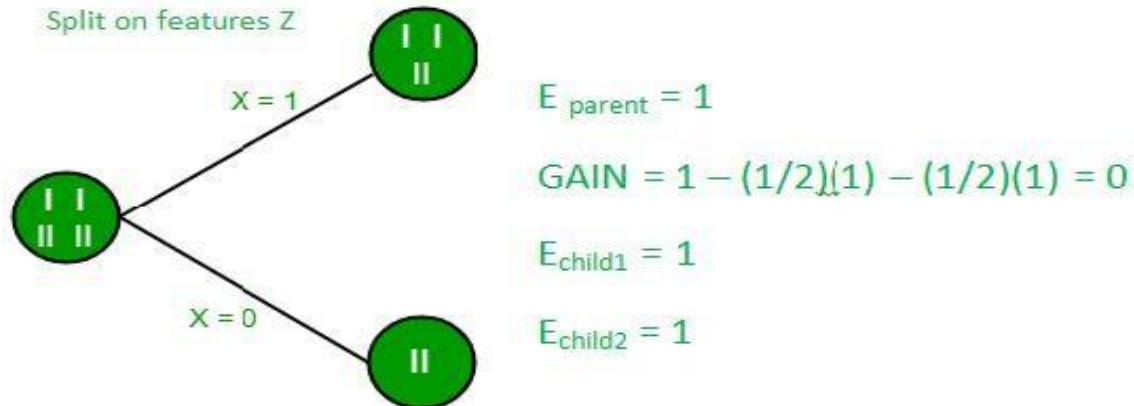
To build a decision tree using Information gain. We will take each of the feature and calculate the information for each feature.



Split on feature X



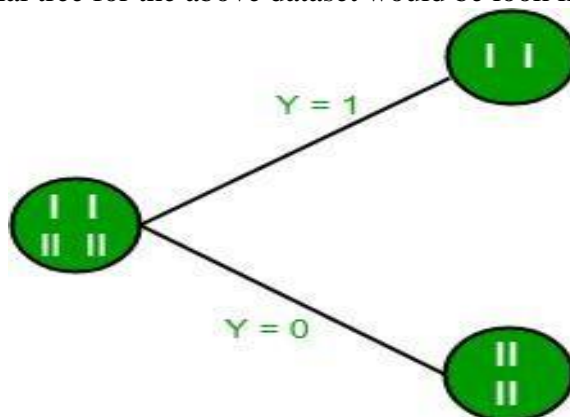
Split on feature Y



Split on feature Z

From the above images we can see that the information gain is maximum when we make a split on feature Y. So, for the root node best suited feature is feature Y. Now we can see that while splitting the dataset by feature Y, the child contains pure subset of the target variable. So we don't need to further split the dataset.

The final tree for the above dataset would be look like this:

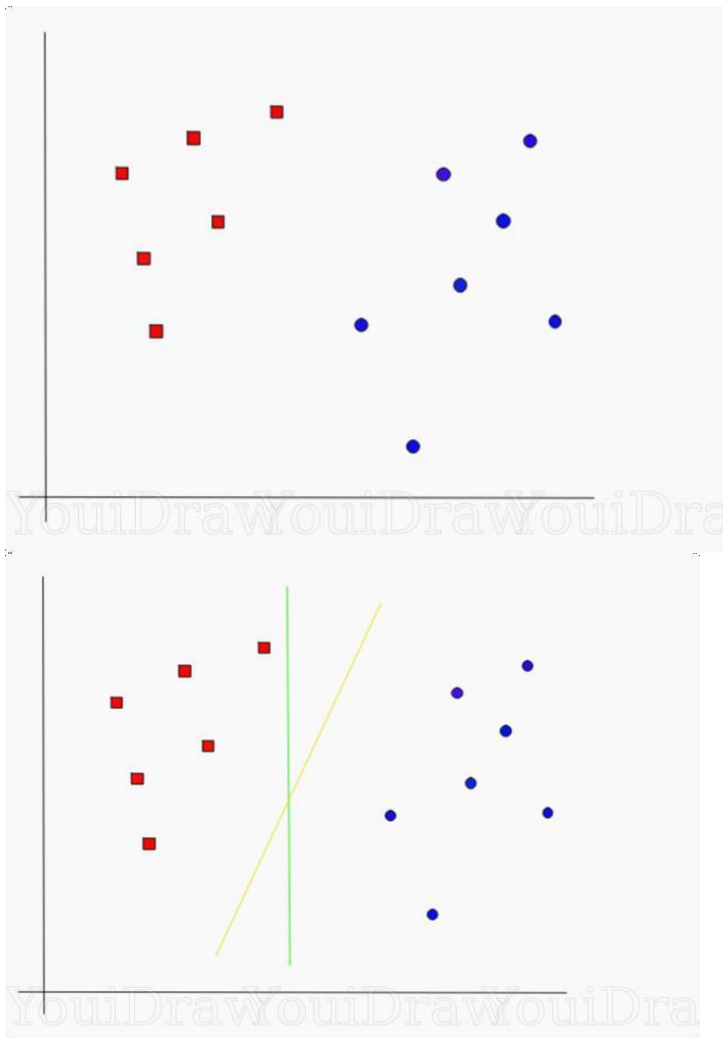


PART-C

Ques 1. Describe the working behavior of support vector machine with diagram.

Ans. SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes. At first approximation what SVMs do is to find a separating line(or hyperplane) between data of two classes. SVM is an algorithm that takes the data as an input and outputs a line that separates those classes if possible.

Lets begin with a problem. Suppose you have a dataset as shown below and you need to classify the red rectangles from the blue ellipses(let's say positives from the negatives). So your task is to find an ideal line that separates this dataset in two classes (say red and blue).

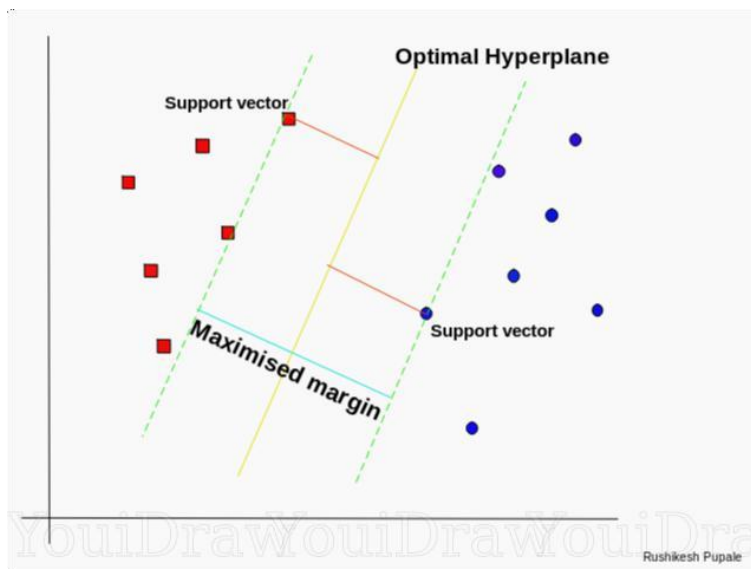


We have two candidates here, the green colored line and the yellow colored line. Which line according to you best separates the data?

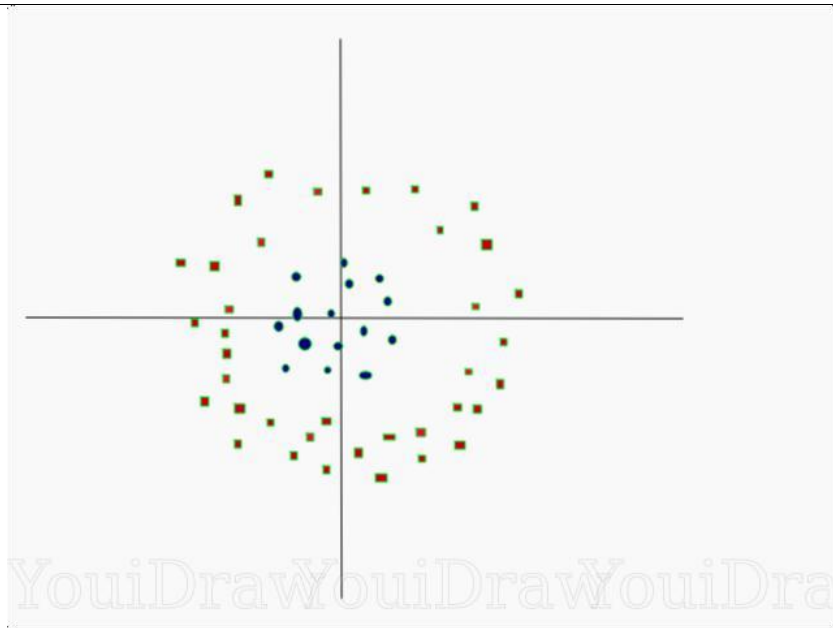
If you selected the yellow line then congrats, because that's the line we are looking for. It's visually quite intuitive in this case that the yellow line classifies better. But, we need something concrete to fix our line. The green line in the image above is quite close to the red class. Though it classifies the current datasets it is not a generalized line and in machine learning our goal is to get a more generalized separator.

SVM's way to find the best line

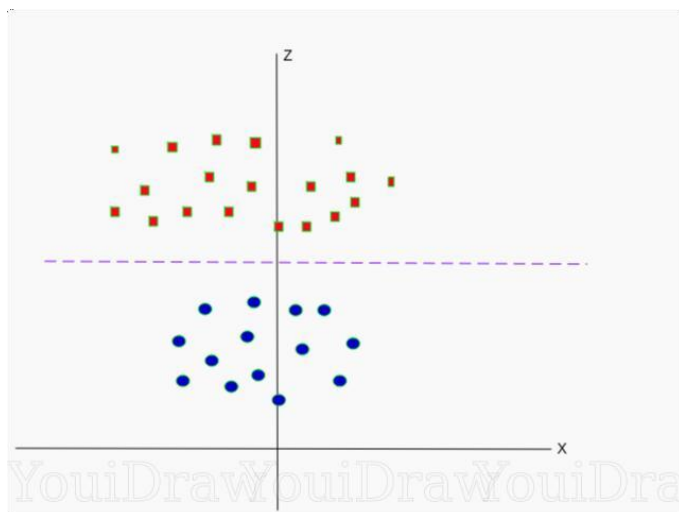
According to the SVM algorithm we find the points closest to the line from both the classes. These points are called support vectors. Now, we compute the distance between the line and the support vectors. This distance is called the margin. Our goal is to maximize the margin. The hyperplane for which the margin is maximum is the optimal hyperplane.



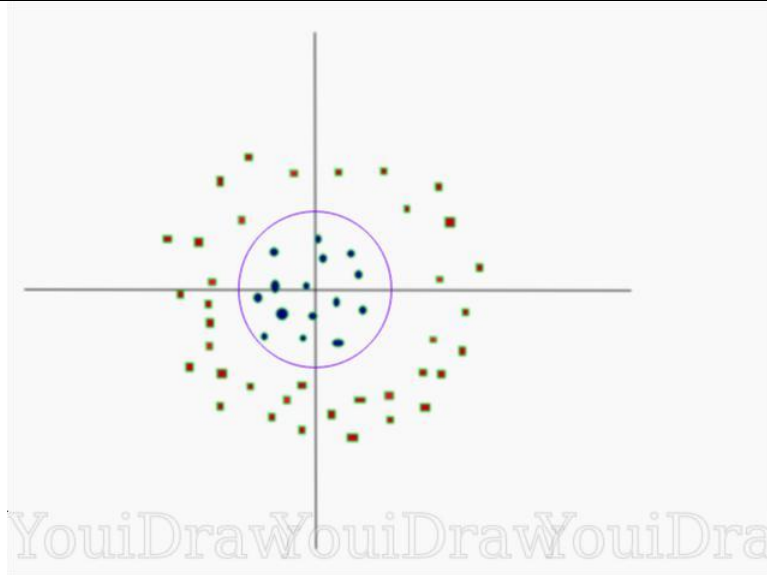
Thus SVM tries to make a decision boundary in such a way that the separation between the two classes (that street) is as wide as possible. Simple, ain't it? Let's consider a bit complex dataset, which is not linearly separable.



This data is clearly not linearly separable. We cannot draw a straight line that can classify this data. But, this data can be converted to linearly separable data in higher dimension. Let's add one more dimension and call it z-axis. Let the co-ordinates on z-axis be governed by the constraint, $z = x^2 + y^2$. So, basically z co-ordinate is the square of distance of the point from origin. Let's plot the data on z-axis.



Now the data is clearly linearly separable. Let the purple line separating the data in higher dimension be $z=k$, where k is a constant. Since, $z=x^2+y^2$ we get $x^2 + y^2 = k$; which is an equation of a circle. So, we can project this linear separator in higher dimension back in original dimensions using this transformation.



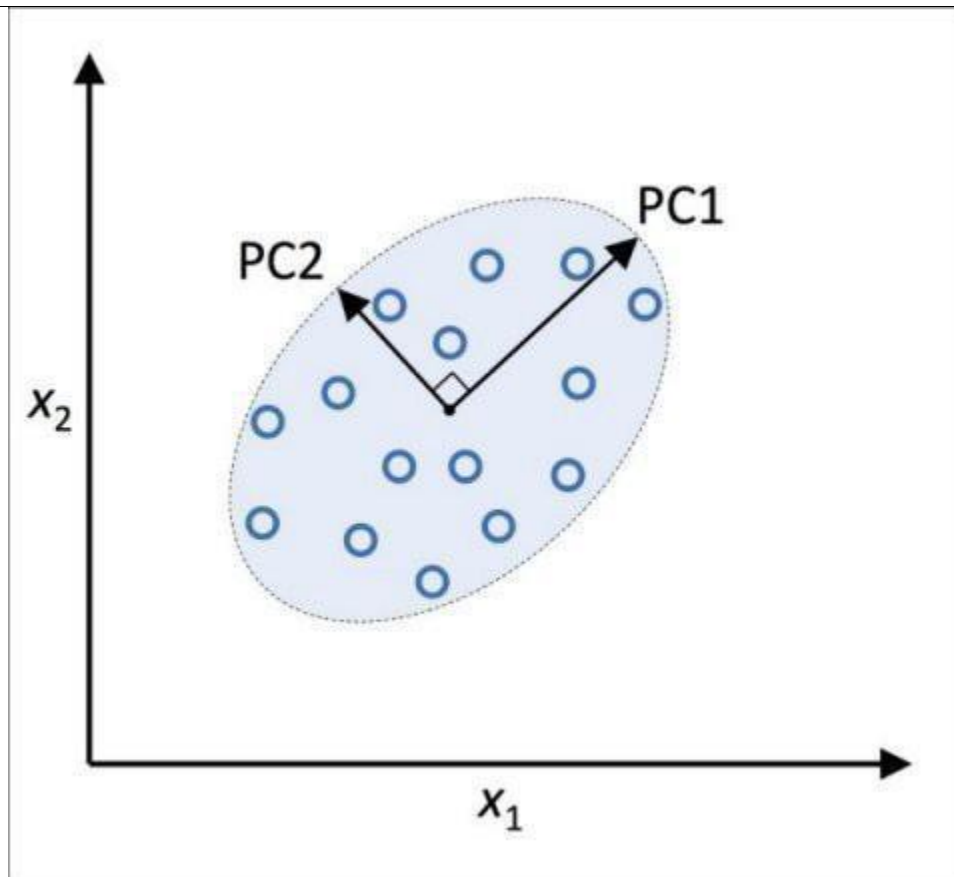
Thus we can classify data by adding an extra dimension to it so that it becomes linearly separable and then projecting the decision boundary back to original dimensions using mathematical transformation. But finding the correct transformation for any given dataset isn't that easy. Thankfully, we can use kernels in sklearn's SVM implementation to do this job.

Ques 2. How Principal Component Analysis is carried out to reduce dimensionality of data sets?

Ans. Principal Component Analysis (PCA) is an unsupervised linear transformation technique that is widely used across different fields, most prominently for feature extraction and dimensionality reduction. Other popular applications of PCA include exploratory data analyses and de-noising of signals in stock market trading, and the analysis of genome data and gene expression levels in the field of bioinformatics.

PCA helps us to identify patterns in data based on the correlation between features. In a nutshell, PCA aims to find the directions of maximum variance in high-dimensional data and projects it onto a new subspace with equal or fewer dimensions than the original one.

The orthogonal axes (**principal components**) of the new subspace can be interpreted as the directions of maximum variance given the constraint that the new feature axes are orthogonal to each other, as illustrated in the following figure:



In the preceding figure, x_1 and x_2 are the original feature axes, and **PC1** and **PC2** are the principal components.

If we use PCA for dimensionality reduction, we construct a $d \times k$ -dimensional transformation matrix \mathbf{W} that allows us to map a sample vector \mathbf{x} onto a new k -dimensional feature subspace that has fewer dimensions than the original d -dimensional feature space:

$$\begin{aligned} \mathbf{x} &= [x_1, x_2, \dots, x_d], & \mathbf{x} &\in \mathbb{R}^d \\ &\downarrow \mathbf{x}\mathbf{W}, & \mathbf{W} &\in \mathbb{R}^{d \times k} \\ \mathbf{z} &= [z_1, z_2, \dots, z_k], & \mathbf{z} &\in \mathbb{R}^k \end{aligned}$$

As a result of transforming the original d -dimensional data onto this new k -dimensional subspace (typically $k \ll d$), the first principal component will have the largest possible variance, and all consequent principal components will have the largest variance given the constraint that these components are uncorrelated (orthogonal) to the other principal components — even if the input features are correlated, the resulting principal components will be mutually orthogonal (uncorrelated).

let's summarize the approach in a few simple steps:

1. Standardize the d -dimensional dataset.
2. Construct the covariance matrix.

3. Decompose the covariance matrix into its eigenvectors and eigenvalues.
4. Sort the eigenvalues by decreasing order to rank the corresponding eigenvectors.
5. Select k eigenvectors which correspond to the k largest eigenvalues, where k is the dimensionality of the new feature subspace ($k \leq d$).
6. Construct a projection matrix \mathbf{W} from the “top” k eigenvectors.
7. Transform the d -dimensional input dataset \mathbf{X} using the projection matrix \mathbf{W} to obtain the new k -dimensional feature subspace.

Ques.4 . Explain the Q function and Q Learning Algorithm assuming deterministic rewards and actions with example.

Ans. Q-learning is the most notable representative of value iteration based methods. Here the goal is to compute directly the optimal value function. These schemes are typically off-policy methods – learning the optimal value function can take place under any policy (subject to exploration requirements).

Recall the definition of the (optimal) Q-function:

$$Q(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) V^*(s').$$

The optimality equation is then $V^*(s) = \max_a Q(s, a)$, $s \in S$, or in terms of Q only:

$$Q(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) \max_{a'} Q(s', a'), \quad s \in S, a \in A$$

The value iteration algorithm is given by:

$$V_{n+1}(s) = \max_a \{ r(s, a) + \gamma \sum_{s'} p(s'|s, a) V_n(s') \}, \quad s \in S$$

with $V_n \rightarrow V^*$. This can be reformulated as

$$\sum$$

$$Q_{+1}(s) = Q(s) + \gamma \left(\max_{a'} Q(s, a') - Q(s) \right),$$

$$s^J \quad a^J$$

with $Q_n \rightarrow Q$.

We can now define the on-line (learning) version of the Q-value iteration equation.

The Q-learning algorithm:

–initialize \hat{Q} .

–At stage n: Observe (s_n, a_n, r_n, s_{n+1}) , and let

$$\begin{aligned} \hat{Q}(s_n, a_n) &:= (1 - \alpha_n) \hat{Q}(s_n, a_n) + \alpha_n [r_n + \gamma \max_{a^J} \hat{Q}(s_{n+1}, a^J)] \\ &= \hat{Q}(s_n, a_n) + \alpha_n [r_n + \gamma \max_{a^J} \hat{Q}(s_{n+1}, a^J) - \hat{Q}(s_n, a_n)] . \end{aligned}$$

The algorithm is obviously very similar to the basic TD schemes for policy evaluation, except for the maximization operation.

Convergence: If all (s, a) pairs are visited i.o., and $\alpha_n \searrow 0$ at appropriate rate, then

$$\hat{Q}_n \rightarrow Q^* .$$

Policy Selection:

– Since learning of Q^* does not depend on optimality of the policy used, we can focus on exploration during learning. However, if learning takes place while the system is in actual operation, we may still need to use a close-to-optimal policy, while using the standard exploration techniques (s -greedy, softmax, etc.).

– When learning stops, we may choose a greedy policy:

$$\hat{\pi}(s) = \max_a \hat{Q}(s, a) .$$

Performance: Q -learning is very convenient to understand and implement; however, convergence may be slower than actor-critic ($TD(\lambda)$) methods, especially if in the latter we only need to evaluate V and not Q .

Q-functions and their Evaluation:

For policy improvement, what we require is actually the Q -function $Q^\pi(s, a)$, rather than $V^\pi(s)$. Indeed, recall the policy-improvement step of policy iteration, which defines the improved policy $\hat{\pi}$ via:

$$\hat{\pi}(s) \in \operatorname{argmax}_{a'} \{ r(s, a) + \gamma \sum_{s'} p(s'|s, a) V^\pi(s') \} \equiv \operatorname{argmax}_{a'} Q^\pi(s, a).$$

How can we estimate Q^π ?

1. Using \hat{V}^π : If we know the one-step model parameters r and p , we may estimate \hat{V}^π as above and compute

$$\hat{Q}^\pi(s, a) \triangleq r(s, a) + \gamma \sum_{s'} p(s'|s, a) \hat{V}^\pi(s').$$

When the model is not known, this requires to estimate r and p on-line.

2. Direct estimation of Q^π : This can be done the same methods as outlined for \hat{V}^π , namely Monte-Carlo or TD methods.

Note that:

- The estimated policy π must be the one used (“on-policy” scheme).
- More variables are estimated in Q than in V .

Ques 5.Elaborate on the types of machine learning and discuss the components in design of a learning system.

Ans. At a high-level, machine learning is simply the study of teaching a computer program or algorithm how to progressively improve upon a set task that it is given. On the research-side of things, machine learning can be viewed through the lens of theoretical and mathematical modeling of how this process works. However, more practically it is the study of how to build applications that exhibit this iterative improvement. There are many ways to frame this idea, but largely there are three major recognized categories: supervised learning, unsupervised learning, and reinforcement learning.

TYPES OF MACHINE LEARNING

1. Supervised Learning: Supervised learning is the most popular paradigm for machine learning. It is the easiest to understand and the simplest to implement. It is very similar to teaching a child with the use of flash cards.

Supervised learning is often described as task-oriented because of this. It is highly focused on a singular task, feeding more and more examples to the algorithm until it can accurately perform on that task. This is the learning type that you will most likely encounter.

2. Unsupervised Learning: Unsupervised learning is very much the opposite of supervised learning. It features no labels. Instead, our algorithm would be fed a lot of data and given the tools to understand the properties of the data. From there, it can learn to group, cluster, and/or organize the data in a way such that a human (or other intelligent algorithm) can come in and make sense of the newly organized data.
3. Reinforcement Learning: Reinforcement learning is fairly different when compared to supervised and unsupervised learning. Where we can easily see the relationship between supervised and unsupervised (the presence or absence of labels), the relationship to reinforcement learning is a bit murkier. Some people try to tie reinforcement learning closer to the two by describing it as a type of learning that relies on a time-dependent sequence of labels, however, my opinion is that that simply makes things more confusing.

Three key components:

1. The exact type of knowledge to be learned (Choosing the Target Function)
2. A representation for this target knowledge (Choosing a representation for the Target Function)
3. A learning mechanism (Choosing an approximation algorithm for the Target Function)

Choose Target Function

Let's take the example of a checkers-playing program that can generate the legal moves (**M**) from any board state (**B**). The program needs only to learn how to choose the best move from among these legal moves. Let's assume a function **NextMove** such that:

NextMove: **B** -> **M**

Here, **B** denotes the set of board states and **M** denotes the set of legal moves given a board state. **NextMove** is our target function.

Choose the Representation of Target Function

We need to choose a representation that the learning algorithm will use to describe the function NextMove. The function NextMove will be calculated as a linear combination of the following board features:

- **x1**: the number of black pieces on the board
- **x2**: the number of red pieces on the board
- **x3**: the number of black kings on the board
- **x4**: the number of red kings on the board
- **x5**: the number of black pieces threatened by red (i.e., which can be captured on red's next turn)
- **x6**: the number of red pieces threatened by black

$$\text{NextMove} = u_0 + u_1x_1 + u_2x_2 + u_3x_3 + u_4x_4 + u_5x_5 + u_6x_6$$

Here u_0, u_1 up to u_6 are the coefficients that will be chosen(learned) by the learning algorithm.

Choose a Function Approximation Algorithm

To learn the target function NextMove, we require a set of training examples, each describing a specific board state **b** and the training value (Correct Move) **y** for **b**. The training algorithm learns/approximate the coefficients u_0, u_1 up to u_6 with the help of these training examples by estimating and adjusting these weights.

We will explore the different ways to find the coefficient u_0, u_1 up to u_6 in the next blog. In the meanwhile think of any learning problem and try to find out a suitable Target function Representation for that.