Machine Learning
Subject Code: 20A05602T

UNIT III - Supervised Learning: Classification - 3-2-1

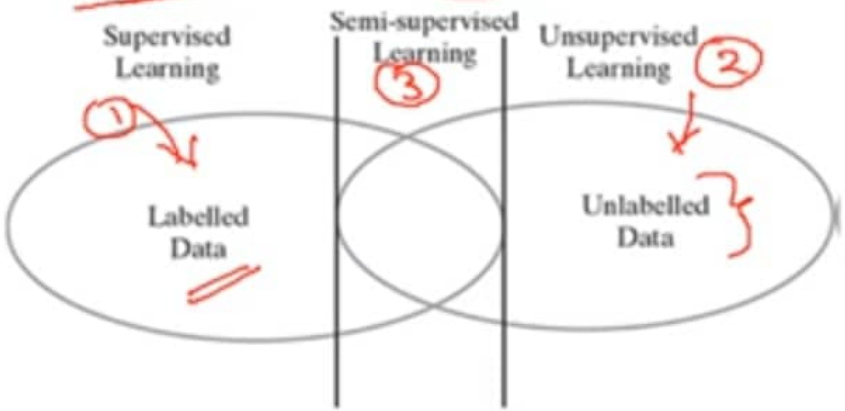
Supervised Learning - Introduction

- Types of Machine Learning
- Example of Supervised Learning
- Classification Model
- Classification Learning Steps

Types of Machine Learning

- There are three types of Machine Learning
1. Supervised learning - Training data is the past information with known value of class field or 'label' ('training data is labelled').
2. Unsupervised Learning - there is no labelled training data.
3. Semi-supervised learning, uses a small amount of labelled data along with unlabelled data for training.

RL



EXAMPLE OF SUPERVISED LEARNING

- Supervised learning is the process of learning from the training data by a machine.
- It can be related to a teacher supervising the learning process of a student who is new to the subject.
- Here, the teacher is the training data.
- In supervised learning, the labelled training data provide the basis for learning, and also the experience or prior knowledge or belief.

Example

- In a hospital, many patients are treated in the general wards.
- The number of beds in the Intensive Care Unit (ICU) is much less.
- So the hospital management moved the patients in the general ward to ICU, when they suddenly worsens
- Without previous planning and preparations, such a spike in demand becomes difficult for the hospital to manage.
- This problem can be addressed that if it is possible to predict which of the patients in the normal wards have a possibility of their health condition worsening and thus need to be moved to ICU.
- This kind of prediction problem comes under the view of supervised learning and under classification.

Example...

- The hospital already has all past patient records.
- The records of the patients whose health condition aggravated in the past and had to be moved to ICU can form the training data for this prediction problem.
- Test results of newly admitted patients are used to classify them as high-risk or low-risk patients.

Example...

- Some more examples of supervised learning are as follows:
  1. Prediction of results of a game based on the past analysis of results
  2. Predicting whether a tumour is malignant or benign on the basis of the analysis of data
  3. Price prediction in domains such as real estate, stocks, etc.

CLASSIFICATION MODEL

- Let us consider two examples, say
  1. 'predicting whether a tumour is malignant or non-malignant' and
  2. 'price prediction in the domain of real estate'.
- both are the problems related to prediction.

1. The tumour prediction, we are trying to predict which category or class, i.e. 'malignant' or 'non-malignant', an unknown input data related to tumour belongs to.
2. The price prediction, trying to predict an absolute value and not a class.

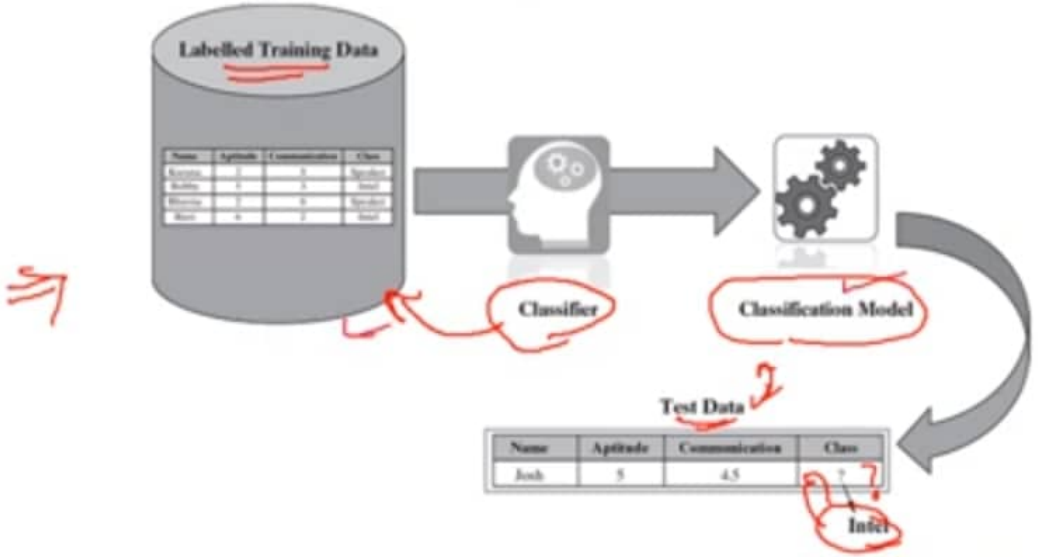
- The problem to predict a categorical or nominal variable, then it is known as a classification problem.

CLASSIFICATION MODEL...

- Classification algorithm is used to identify the category of new data on the basis of training data
- In classification, a program learn from given dataset (training data) then classify new data (test data) into number of classes or groups.
  - Yes/No, Cat/Dog, Red/Green/Blue, Spam/not spam etc.
- The classes can be called as targets or labels or categories.

CLASSIFICATION MODEL...

- A classification model is obtained from the labelled training data by a classifier algorithm.
- On the basis of the model, a class label (e.g. 'Intel' as in the case of the test data) is assigned to the test data.



CLASSIFICATION MODEL...

- A critical classification problem in the context of the banking domain is identifying potentially fraudulent transactions.
- Because there are millions of transactions which have to be scrutinized to identify whether a particular transaction might be a fraud transaction,
- it is not possible for any human being to carry out this task.
- Machine learning solves this problem efficiently, on the basis of the past transaction data, labelled as fraudulent, all new incoming transactions are marked or labelled as usual or suspicious.
- The suspicious transactions are subsequently segregated for a closer review.

CLASSIFICATION MODEL...

- Some typical classification problems include the following:
- Image classification
- Disease prediction
- Win–loss prediction of games
- Prediction of natural calamity such as earthquake, flood, etc.
- Handwriting recognition

CLASSIFICATION LEARNING STEPS

1. Problem Identification:
2. Identification of Required Data:
3. Data Pre-processing:
4. Definition of Training Data Set:
5. Algorithm Selection:
6. Training:
7. Evaluation with the Test Data Set:

Problem Identification

- Identifying the problem is the first step in the supervised learning model
- The problem needs to be a well-formed problem;
- i.e. a problem with well-defined goals and benefit, which has a long-term impact.

Identification of Required Data:

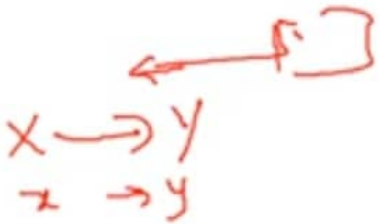
- On the basis of the problem identified above, the required data set that exactly represents the identified problem needs to be evaluated.
- For example: If the problem is to predict whether a tumour is malignant or not,
- then the corresponding patient data sets related to malignant tumour and normal tumours are to be identified.

Data Pre-processing:

- The data is gathered from different sources, it is usually collected in a raw format and is not ready for immediate analysis.
- Data pre-processing refers to the transformations applied to the identified data before feeding the same into the algorithm.
- This is related to the cleaning/transforming the data set.
- This step ensures that all the unnecessary/irrelevant data elements are removed.
- And the data is ready to be fed into the machine learning algorithm.

Definition of Training Data Set:

- Before starting the analysis, the user should decide what kind of data set is to be used as a training set.
- a set of 'input meta-objects' and corresponding 'output meta-objects' are gathered.
- Thus, a set of data input (X) and corresponding outputs (Y) is gathered either from human experts or experiments.
- The training set needs to be actively representative of the real-world use of the given scenario.

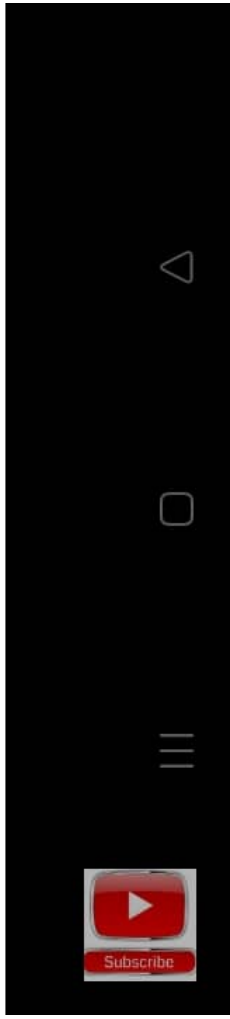


Algorithm Selection:

- This is the most critical step of supervised learning model.
- This involves determining the structure of the learning function and the corresponding learning algorithm.
- On the basis of various parameters, the best algorithm for a given problem is chosen.

Training:

- The identified learning algorithm will run on the gathered training set, with the required control parameters as input to the algorithm
- These parameters (inputs given to algorithm) may also be adjusted by optimizing performance on a subset (called as validation set)



Machine Learning
Subject Code: 20A05602T

UNIT III - Supervised Learning: Classification - 3-2-2

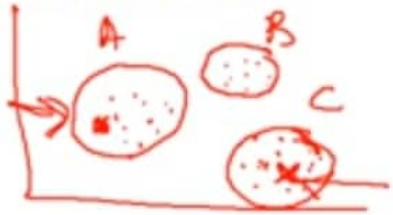
k-Nearest Neighbour (kNN) Algorithm

- k-Nearest Neighbour (kNN)
- Solved Problem
- Problems in selecting k value
- The kNN is a lazy learner
- Applications of kNN

Name	Aptitude	Communication	Class
Karuna	2	5	Speaker
Bhuvna	2	6	Speaker
Gaurav	7	6	Leader
Parul	7	2.5	Intel
Dinesh	8	6	Leader
Jani	4	7	Speaker
Bobby	5	3	Intel
Parimal	3	5.5	Speaker
Govind	8	3	Intel
Susant	6	5.5	Leader
Gouri	6	4	Intel
Bharat	6	7	Leader
Ravi	6	2	Intel
Pradeep	9	7	Leader
Josh	5	4.5	Intel

k-Nearest Neighbour (kNN) Algorithm

- The kNN algorithm is a simple but extremely powerful supervised learning algorithm.
- K-NN algorithm
  - stores all the available data and
  - classifies a new data point based on the similarity.
- The kNN is used in classifications or predictions, the grouping of an individual data point.
- This means when new data appears, then it can be easily classified into a well suite Category/Class.



kNN algorithm

- Input: Training data set, test data set (or data points), value of 'k' (i.e. number of nearest neighbours to be considered)

- Steps:
- Do for all test data points
  - Calculate the distance (usually Euclidean distance) of the test data point from the different training data points.
  - Find the closest 'k' training data points, i.e. training data points whose distances are least from the test data point.
  - If k = 1
    - Then assign class label of the training data point to the test data point
  - Else
    - Whichever class label is mostly present in the training data points, assign that class label to the test data point
  - End do

Example Problem - kNN Algorithm

- Let us try to understand the algorithm with a simple data set.
- A Student data set consists of 15 students, studying in a class.
- Each of the students has been assigned a score on a scale of 10 on two performance parameters –
- 'Aptitude' and 'Communication'.
- Also, a **class value** is assigned to each student based on the following criteria:
- 1. good communication skills & good level of aptitude have been classified as 'Leader'
- 2. good communication skills but not so good level of aptitude have been classified as 'Speaker'
- 3. not so good communication skill but a good level of aptitude have been classified as 'Intel'

Name	Aptitude	Communication	Class
Karuna	2	5	Speaker
Bhuvna	2	6	Speaker
Gaurav	7	6	Leader
Parul	7	2.5	Intel
Dinesh	8	6	Leader
Jani	8	7	Speaker
Bobby	5	3	Intel
Parimal	3	5.5	Speaker
Govind	8	3	Intel
Susant	6	5.5	Leader
Gouri	6	4	Intel
Bharat	6	7	Leader
Ravi	6	2	Intel
Pradeep	9	7	Leader
Josh	5	4.5	Intel

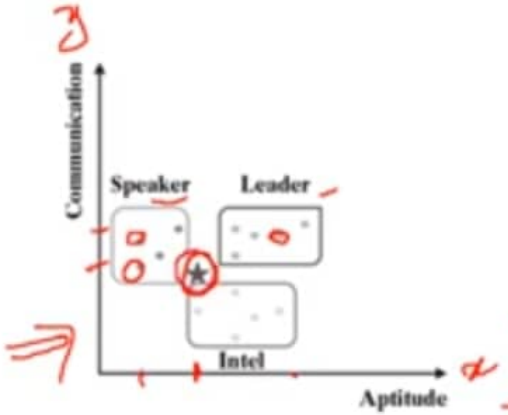
- To build a classification model, a part of the labelled input data is retained as test data.
- The remaining portion of the input data is used to train the model – hence known as training data.
- The test data is used to evaluate the performance of the model.
- the record of the student named Josh is assumed to be the test data.

Name	Aptitude	Communication	Class
Karuna	2	5	Speaker
Bhuvna	2	6	Speaker
Gaurav	7	6	Leader
Parul	7	2.5	Intel
Dinesh	8	6	Leader
Jani	4	7	Speaker
Bobby	5	3	Intel
Parimal	3	5.5	Speaker
Govind	8	3	Intel
Susant	6	5.5	Leader
Gouri	6	4	Intel
Bharat	6	7	Leader
Ravi	6	2	Intel
Pradeep	9	7	Leader
Josh	5	4.5	Intel

Training Data

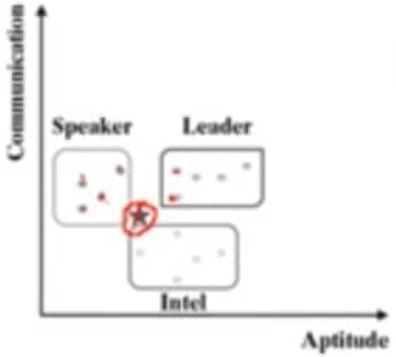
Test Data

- considering the features 'Aptitude' and 'Communication' can be represented as dots in a two-dimensional feature space.
- the training data points having the same class value are coming close to each other.
- The feature 'Name' is ignored because, as we can understand, it has no role to play in deciding the class value.
- * in the diagram is test data set



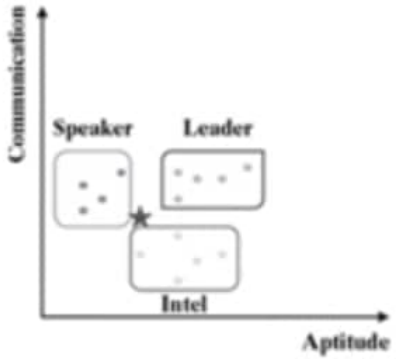
Name	Aptitude	Communication	Class
Karuna	2	5	Speaker
Bhuvna	2	6	Speaker
Gaurav	7	6	Leader
Parul	7	2.5	Intel
Dinesh	8	6	Leader
Jani	4	7	Speaker
Bobby	5	3	Intel
Parimal	3	5.5	Speaker
Govind	8	3	Intel
Susant	6	5.5	Leader
Gouri	6	4	Intel
Bharat	6	7	Leader
Ravi	6	2	Intel
Pradeep	9	7	Leader
Josh	5	4.5	???

- To find the nearest neighbours of the test data point, Euclidean distance of the different dots need to be calculated from the asterisk.
- If k = 1, only the closest training data element is considered.
- If k=3, only three nearest neighbours or three training data elements closest to the test data element are considered.
- The class label of that data element is directly assigned to the test data element.



Name	Aptitude	Communication	Class	Distance	k = 1	k = 2	k = 3
Karuna	2	5	Speaker	3.041			
Bhuvna	2	6	Speaker	3.354			
Parimal	3	5.5	Speaker	2.236			
Jani	4	7	Speaker	2.693			
Bobby	5	3	Intel	1.500			1.500
Ravi	6	2	Intel	2.693			
Gouri	6	4	Intel	1.118	1.118	1.118	1.118
Parul	7	2.5	Intel	2.828			
Govind	8	3	Intel	3.354			
Susant	6	5.5	Leader	1.414			
Bharat	6	7	Leader	2.693			
Gaurav	7	6	Leader	2.500			
Dinesh	8	6	Leader	3.354			
Pradeep	9	7	Leader	4.717			
Josh	5	4.5	???				

- Gouri and Bobby have class value 'Intel', while Susant has class value 'Leader'.
- In this case, the class value of Josh is decided by majority voting.
- Because the class value of 'Intel' is formed by the majority of the neighbours, the class value of Josh is assigned as 'Intel'.
- This same process can be extended for any value of k.



K=3

Name	Aptitude	Communication	Class	Distance	k = 1	k = 2	k = 3
Karuna	2	5	Speaker	3.041			
Bhuvna	2	6	Speaker	3.354			
Parimal	3	5.5	Speaker	2.236			
Jani	4	7	Speaker	2.693			
Bobby	5	3	Intel	1.500			1.500
Ravi	6	2	Intel	2.693			
Gouri	6	4	Intel	1.118	1.118	1.118	1.118
Parul	7	2.5	Intel	2.828			
Govind	8	3	Intel	3.354			
Susant	6	5.5	Leader	1.414			
Bharat	6	7	Leader	2.693			
Gaurav	7	6	Leader	2.500			
Dinesh	8	6	Leader	3.354			
Pradeep	9	7	Leader	4.717			
Josh	5	4.5	???				

Intel

kNN Algorithm – Selecting k value

- If the value of k is very large (in the extreme case equal to the total number of records in the training data), the class label of the majority class of the training data set will be assigned to the test data regardless of the class labels of the neighbours nearest to the test data.
- If the value of k is very small (in the extreme case equal to 1), the class value of a noisy data or outlier in the training data set which is the nearest neighbour to the test data will be assigned to the test data.
- The best k value is somewhere between these two extremes.

kNN Algorithm – Selecting k value

- Few strategies are adopted by machine learning practitioners to arrive at a value for k.
- 1. k equal to the square root of the number of training records.
- 2. test several k values on a variety of test data sets and choose the one that delivers the best performance.
- 3. choose a larger value of k, but apply a weighted voting process in which the vote of close neighbours is considered more influential than the vote of distant neighbours.

Why the kNN algorithm is called a lazy learner?

- The eager learners follow the general steps of machine learning,
- i.e. perform an abstraction of the information obtained from the input data and then follow it through by a generalization step.
- In the kNN algorithm, these steps are completely skipped.
- It stores the training data and directly applies the philosophy of nearest neighbourhood finding to arrive at the classification.
- there is no learning happening in the real sense.
- Therefore, kNN falls under the category of lazy learner.

Strengths and weakness of the kNN algorithm

- Strengths
- Extremely simple algorithm – easy to understand
- Very effective in certain situations,
- Very fast or almost no time required for the training phase
- Weaknesses
- Does not learn anything in the real sense.
- Classification is done completely on the basis of the training data.
- If the training data does not represent the problem domain systematically, the algorithm fails to make an effective classification.
- Also, a large amount of computational space is required to load the training data for classification.

Application of the kNN algorithm

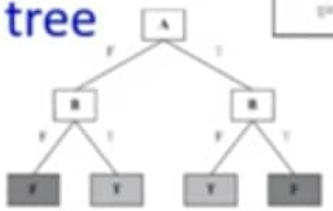
- The recommender systems, recommend users, with different items which are similar to a particular item that the user seems to like.
- The liking pattern may be revealed from past purchases or browsing history and the similar items are identified using the kNN algorithm.
- Information retrieval (concept search), Searching documents/ contents similar to a given document/content.

Machine Learning
Subject Code: 20A05602T

UNIT III - Supervised Learning: Classification - 3-2-3

Decision tree Algorithm

- Decision Tree
- Building a decision tree
- A Decision Tree – Example – Car Driving
- Avoiding overfitting in decision tree
- Strengths and Weakness of decision tree
- Application of decision tree

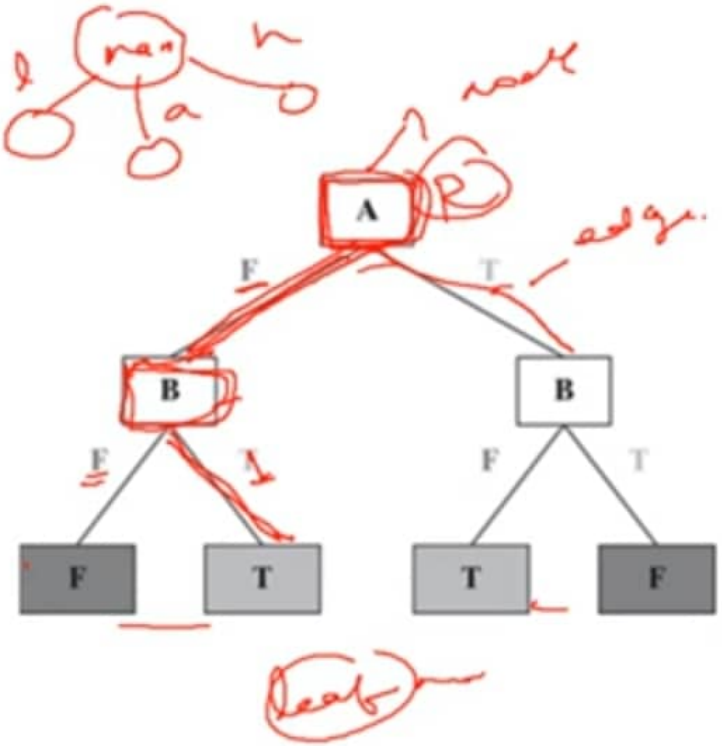


Decision Tree

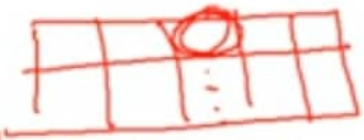
- Decision tree learning is one of the most widely adopted algorithms for classification, and it builds a model in the form of a tree structure.
- A decision tree is used for multi-dimensional analysis with multiple classes.
- It is characterized by fast execution time and ease in the interpretation of the rules.
- The goal of decision tree learning is **to create a model (based on the past data called past vector) that predicts the value of the output variable based on the input variables in the feature vector.**

A Decision Tree...

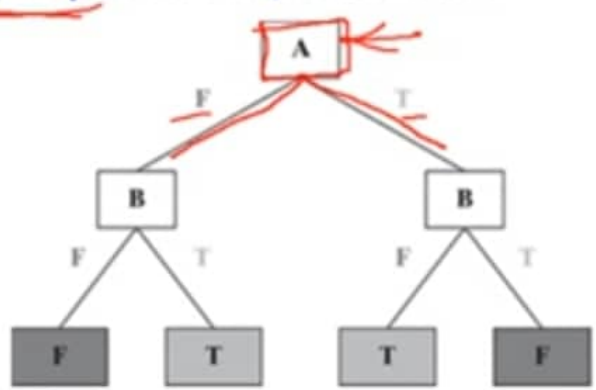
- A decision tree consists of three types of nodes:
  - Root Node, Branch Node, Leaf Node
- The first node is called as 'Root' Node.
- 'B' is the Branch Node. 'T' & 'F' are Leaf Nodes
- Each internal node tests an attribute (represented as A/B within the boxes).
- Each branch corresponds to an attribute value (T/F).
- The output variable is determined by the path that starts at the root to leaf node based on attribute values of each corresponding nodes.



Building a Decision Tree

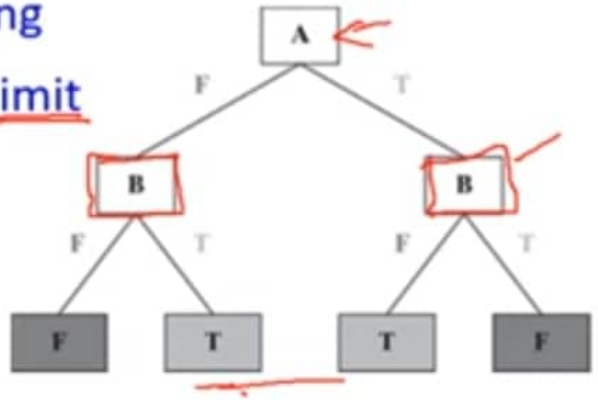


- Decision trees are built based on the training data by implementing recursive partitioning.
- It splits the data into multiple subsets on the basis of the feature values.
- First it selects the root node, i.e. **select one feature from the training data set which predicts the target class in the strongest way.**
- The decision tree splits the data set into multiple partitions, based on the class values of selected feature.
- This is the first set of branches.



Building a decision tree...

- Likewise, the algorithm continues splitting the nodes on the basis of the feature which helps in the best partition. untill a stopping criterion is reached.
- The usual stopping criteria are –
- 1. All or most of the examples at a particular node have the same class
- 2. All features have been used up in the partitioning
- 3. The tree has grown to a pre-defined threshold limit



A Decision Tree – Example – Car Driving

- The decision to be taken is whether to 'Keep Going' or to 'Stop', depends on various situations
- If the signal is RED in colour, then the car should be stopped.
- If there is not enough gas (petrol) in the car, the car should be stopped at the next available gas station.

Algorithm for decision tree

- Input: Training data set, test data set (or data points)
- Steps:
- Do for all attributes
- Calculate the entropy E of the attribute F
- if Ei < Emin
- then Emin = Ei and Fmin = Fi
- end if
- End do
- Split the data set into subsets using the attribute Fmin
- Draw a decision tree node containing the attribute Fmin and split the data set into subsets
- Repeat the above steps until the full tree is drawn covering all the attributes of the original table.

Avoiding overfitting in Decision Tree – Pruning

- If the decision tree algorithm is applied until the stopping criterion, may keep growing indefinitely – and this results in overfitting problem.
- To prevent this, pruning of the decision tree is essential.
  - It reduces the size of the tree such that
  - Then the model will be more generalized and
  - It can classify unknown and unlabelled data in a better way.

Avoiding overfitting in Decision Tree – Pruning...

- There are two approaches of pruning:
- Pre-pruning: the tree is stopped from further growing once it reaches a certain number of decision nodes or decisions.
- Hence, in this strategy, the algorithm avoids overfitting as well as optimizes computational cost, But a chance to ignore important information
- Post-pruning: Allow the tree to grow entirely and then post-prune some of the branches from it.

Strengths of decision tree

- It produces very simple understandable rules.
- For smaller trees, not much mathematical and computational knowledge is required to understand this model.
- Works well for most of the problems.
- It can handle both numerical and categorical variables.
- Can work well both with small and large training data sets.
- Decision trees provide a definite clue of which features are more useful for classification.

Weaknesses of decision tree

- Decision tree models are often biased towards features having more number of possible values, i.e. levels.
- This model gets overfitted or underfitted quite easily.
- Decision trees are prone to errors in classification problems with many classes and relatively small number of training examples.
- A decision tree can be computationally expensive to train.
- Large trees are complex to understand.

Application of decision tree

1. Business Management - extract useful information from databases
2. Customer Relationship Management - decision trees are useful for understanding their customers' needs and preferences
3. Fraudulent Statement Detection - detection of Fraudulent Financial Statements (FFS) - discover all hidden information
4. Engineering – widely used in energy consumption and fault diagnosis
   - Energy Consumption - Energy consumption concerns how much electricity has been used by individuals
   - Fault Diagnosis - the identification of a faulty parts/tools in machineries
   - Healthcare Management - to discover and/or explore hidden information of patients illness

Machine Learning
Subject Code: 20A05602T
UNIT III - Supervised Learning: Classification - 3-2-4
Decision Tree Algorithm

- Decision Tree
- Solved Problem
- Searching a Decision Tree
  - Exhaustive search
  - Branch and bound search
- Entropy of a Decision Tree
- Information gain of a Decision Tree

CGPA	Communication	Aptitude	Programming Skill	Job offered?
High	Good	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	Low	Good	No
Low	Good	Low	Bad	No
High	Good	High	Bad	Yes
High	Good	High	Good	Yes
Medium	Bad	Low	Bad	No
Medium	Bad	Low	Good	No
High	Bad	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	High	Bad	No
Low	Bad	High	Bad	No
Medium	Good	High	Bad	Yes
Low	Good	Low	Good	No
High	Bad	Low	Bad	No
Medium	Bad	High	Good	No
High	Bad	Low	Bad	No
Medium	Good	High	Bad	Yes

GST – Interview Evaluation

- Global Technology Solutions (GTS), is coming to College of Engineering and Management (CEM) for hiring B.E./B.Tech. students.
- Data set for students interview evaluation results
- Attribute details
- CGPA-
  - High, Medium, Low
- Communication
  - Good, Bad
- Aptitude
  - High, Low
- Programming Skill
  - Good, Bad
- Job Offered? (target variable)
  - Yes, No

CGPA	Communication	Aptitude	Programming Skill	Job offered?
High	Good	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	Low	Good	No
Low	Good	Low	Bad	No
High	Good	High	Bad	Yes
High	Good	High	Good	Yes
Medium	Bad	Low	Bad	No
Medium	Bad	Low	Good	No
High	Bad	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	High	Bad	No
Low	Bad	High	Bad	No
Medium	Good	High	Bad	Yes
Low	Good	Low	Good	No
High	Bad	Low	Bad	No
Medium	Bad	High	Good	No
High	Bad	Low	Bad	No
Medium	Good	High	Bad	Yes

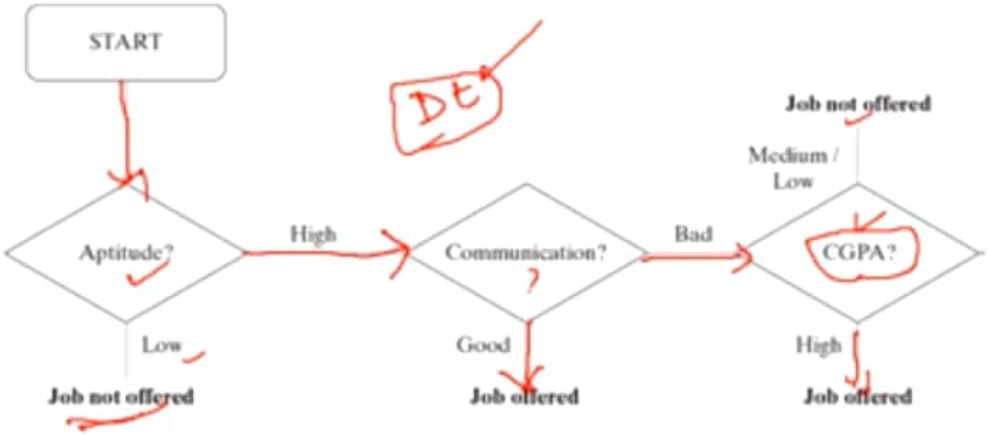
A Decision Tree – Example...

- Problem Statement:
- Chandra, a student of CEM, wants to find out if he may be offered a job in GTS.
- His details are
  - CGPA – high,
  - Communication – Bad;
  - Aptitude – High;
  - Programming skills – Bad
- Let us try to solve this problem, predicting whether Chandra will get a job offer, by using the decision tree model.

CGPA	Communication	Aptitude	Programming Skill	Job offered?
High	Good	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	Low	Good	No
Low	Good	Low	Bad	No
High	Good	High	Bad	Yes
High	Good	High	Good	Yes
Medium	Bad	Low	Bad	No
Medium	Bad	Low	Good	No
High	Bad	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	High	Bad	No
Low	Bad	High	Bad	No
Medium	Good	High	Bad	Yes
Low	Good	Low	Good	No
High	Bad	Low	Bad	No
Medium	Bad	High	Good	No
High	Bad	Low	Bad	No
Medium	Good	High	Bad	Yes

A Decision Tree – Example...

- For Aptitude = High, job offer condition is TRUE for all the cases where Communication = Good.
- For cases where Communication = Bad, job offer condition is TRUE for all the cases where CGPA = High.



CGPA	Communication	Aptitude	Programming Skill	Job offered?
High	Good	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	Low	Good	No
Low	Good	Low	Bad	No
High	Good	High	Bad	Yes
High	Good	High	Good	Yes
Medium	Bad	Low	Bad	No
Medium	Bad	Low	Good	No
High	Bad	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	High	Bad	No
Low	Bad	High	Bad	No
Medium	Good	High	Bad	Yes
Low	Good	Low	Good	No
High	Bad	Low	Bad	No
Medium	Bad	High	Good	No
High	Bad	Low	Bad	No
Medium	Good	High	Bad	Yes

Searching a Decision Tree

- By using the decision tree we need to predict whether Chandra might get a job offer for the given parameter values:
  - CGPA = High, Communication = Bad, Aptitude = High, Programming skills = Bad.

Decision Tree Implementation

- There are many implementations of decision tree,
  - C5.0,
  - CART (Classification and Regression Tree),
  - CHAID (Chi-square Automatic Interaction Detector) and
  - ID3 (Iterative Dichotomiser 3) algorithms.
- The biggest challenge of a decision tree algorithm is to find out which feature to split upon.
- Identifying the feature is, that the data should be split, and the split should contain examples (dataitems) belonging to a single class.
- Then the partitions are considered to be pure.

Decision Tree Implementation

1. Entropy is a measure of impurity of an attribute or feature adopted by many algorithms such as ID3 and C5.0.
2. The information gain is calculated on the basis of the decrease in entropy (S) after a data set is split according to a particular attribute (A).

- Constructing a decision tree is finding an attribute that returns the highest information gain.

Entropy of a decision tree

- Let us say S is the sample set of training examples.
- Entropy(S) measuring the impurity of S is defined as

$$\text{Entropy}(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

- where
- c is the number of different class labels and
- p refers to the proportion of values falling into the i-th class label.

Entropy of a decision tree

- In the training data, two values for the target class 'Job Offered?' – Yes and No.
- The value of p for
  - class value 'Yes' is 0.44 (i.e. 8/18) and
  - class value 'No' is 0.56 (i.e. 10/18)
- So, we can calculate the entropy as
- Entropy(S) = -0.44 log (0.44) - 0.56 log (0.56)
- = 0.99.

Training data table (18)

CGPA	Communication	Aptitude	Programming Skill	Job offered?
High	Good	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	Low	Good	No
Low	Good	Low	Bad	No
High	Good	High	Bad	Yes
High	Good	High	Good	Yes
Medium	Bad	Low	Bad	No
Medium	Bad	Low	Good	No
High	Bad	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	High	Bad	No
Low	Bad	High	Bad	No
Medium	Good	High	Bad	Yes
Low	Good	Low	Good	No
High	Bad	Low	Bad	No
Medium	Bad	High	Good	No
High	Bad	Low	Bad	No
Medium	Good	High	Bad	Yes

Information gain of a decision tree

- Information gain for a particular feature A is calculated by the difference in entropy before a split (or Sbs) with the entropy after the split (Sas).
- Information Gain(S, A) = Entropy(Sbs) – Entropy(Sas)
- For calculating the entropy after split, **entropy for all partitions needs to be considered.**
- Then, the weighted summation of the entropy for each partition can be taken as the total entropy after split.
- For performing weighted summation, the proportion of examples falling into each partition is used as weight.

$$\textbf{Entropy}(S_{\text{as}}) = \sum_{i=1}^{n} w_i \, \text{Entropy}\,(p_i)$$

Decision Tree Implementation...

- Let us examine the value of information gain for the training data set.
- find the value of entropy at the beginning before any split happens and then again after the split happens.
- We will compare the values for all the cases –
  - 1. when the feature 'CGPA' is used for the split
  - 2. when the feature 'Communication' is used for the split
  - 3. when the feature 'Aptitude' is used for the split
  - 4. when the feature 'Programming Skills' is used for the split

TDS

CGPA	Communication	Aptitude	Programming Skill	Job offered?
High	Good	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	Low	Good	No
Low	Good	Low	Bad	No
High	Good	High	Bad	Yes
High	Good	High	Good	Yes
Medium	Bad	Low	Bad	No
Medium	Bad	Low	Good	No
High	Bad	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	High	Bad	No
Low	Bad	High	Bad	No
Medium	Good	High	Bad	Yes
Low	Good	Low	Good	No
High	Bad	Low	Bad	No
Medium	Bad	High	Good	No
High	Bad	Low	Bad	No
Medium	Good	High	Bad	Yes

$$\text{Entropy}(S_{as}) = \sum_{i=1}^{n} w_i \, \text{Entropy}(p_i)$$

CGPA	Communication	Aptitude	Programming Skill	Job offered?
High	Good	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	Low	Good	No
Low	Good	Low	Bad	No
High	Good	High	Bad	Yes
High	Good	High	Good	Yes
Medium	Bad	Low	Bad	No
Medium	Bad	Low	Good	No
High	Bad	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	High	Bad	No
Low	Bad	High	Bad	No
Medium	Good	High	Bad	Yes
Low	Good	Low	Good	No
High	Bad	Low	Bad	No
Medium	Bad	High	Good	No
High	Bad	Low	Bad	No
Medium	Good	High	Bad	Yes

(a) Original data set:

	Yes	No	Total
Count	8	10	18
pi	0.44	0.56	
-pi*log(pi)	0.52	0.47	0.99

Total Entropy = 0.99

(b) Splitted data set (based on the feature 'CGPA'):

CGPA = High

	Yes	No	Total
Count	4	2	6
pi	0.67	0.33	
-pi*log(pi)	0.39	0.53	0.92

Total Entropy = 0.69

CGPA = Medium

	Yes	No	Total
Count	4	3	7
pi	0.57	0.43	
-pi*log(pi)	0.46	0.52	0.99

CGPA = Low

	Yes	No	Total
Count	0	5	5
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

Information Gain = 0.30

(c) Splitted data set (based on the feature 'Communication'):

Communication = Good

	Yes	No	Total
Count	7	2	9
pi	0.78	0.22	
-pi*log(pi)	0.28	0.48	0.76

Total Entropy = 0.63

Communication = Bad

	Yes	No	Total
Count	1	8	9
pi	0.11	0.89	
-pi*log(pi)	0.35	0.15	0.50

Information Gain = 0.36

(d) Splitted data set (based on the feature 'Aptitude'):

Aptitude = High

	Yes	No	Total
Count	8	3	11
pi	0.73	0.27	
-pi*log(pi)	0.33	0.51	0.85

Total Entropy = 0.52

Aptitude = Low

	Yes	No	Total
Count	0	7	7
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

Information Gain = 0.47

(e) Splitted data set (based on the feature 'Programming Skill'):

Programming Skill = Good

	Yes	No	Total
Count	5	4	9
pi	0.56	0.44	
-pi*log(pi)	0.47	0.52	0.99

Total Entropy = 0.95

Programming Skill = Bad

	Yes	No	Total
Count	3	6	9
pi	0.33	0.67	
-pi*log(pi)	0.53	0.39	092

Information Gain = 0.04

Decision Tree Implementation...

- As calculated, entropy of the data set before split Entropy(Sbs)) = 0.99,
- entropy of the data set after split. Entropy (Sas)) is
- 0.69 when the feature 'CGPA' is used for split
- 0.63 when the feature 'Communication' is used for split
- 0.52 when the feature 'Aptitude' is used for split
- 0.95 when the feature 'Programming skill' is used for split

(a) Original data set:

	Yes	No	Total
Count	8	10	18
pi	0.44	0.56	
-pi*log(pi)	0.52	0.47	0.99

Total Entropy = 0.99

(b) Splitted data set (based on the feature 'CGPA'):

CGPA = High

	Yes	No	Total
Count	4	2	6
pi	0.67	0.33	
-pi*log(pi)	0.39	0.53	0.92

Total Entropy = 0.69

CGPA = Medium

	Yes	No	Total
Count	4	3	7
pi	0.57	0.43	
-pi*log(pi)	0.46	0.52	0.99

Information Gain = 0.30

CGPA = Low

	Yes	No	Total
Count	0	5	5
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

(c) Splitted data set (based on the feature 'Communication'):

Communication = Good

	Yes	No	Total
Count	7	2	9
pi	0.78	0.22	
-pi*log(pi)	0.28	0.48	0.76

Total Entropy = 0.63

Communication = Bad

	Yes	No	Total
Count	1	8	9
pi	0.11	0.89	
-pi*log(pi)	0.35	0.15	0.50

Information Gain = 0.36

(d) Splitted data set (based on the feature 'Aptitude'):

Aptitude = High

	Yes	No	Total
Count	8	3	11
pi	0.73	0.27	
-pi*log(pi)	0.33	0.51	0.85

Total Entropy = 0.52

Aptitude = Low

	Yes	No	Total
Count	0	7	7
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

Information Gain = 0.47

(e) Splitted data set (based on the feature 'Programming Skill'):

Programming Skill = Good

	Yes	No	Total
Count	5	4	9
pi	0.56	0.44	
-pi*log(pi)	0.47	0.52	0.99

Total Entropy = 0.95

Programming Skill = Bad

	Yes	No	Total
Count	3	6	9
pi	0.33	0.67	
-pi*log(pi)	0.53	0.39	092

Information Gain = 0.04

Decision Tree Implementation...

- the information gain from the feature 'CGPA' = 0.99 – 0.69 = 0.3,
- the information gain from the feature 'Communication' = 0.99 – 0.63 = 0.36.
- the information gain for 'Aptitude'=0.99-0.52= 0.47
- 'Programming skills' = 0.99-0.95= 0.04,

(a) Original data set:

	Yes	No	Total
Count	8	10	18
pi	0.44	0.56	
-pi*log(pi)	0.52	0.47	0.99

Total Entropy = 0.99

(b) Splitted data set (based on the feature 'CGPA'):

CGPA = High

	Yes	No	Total
Count	4	2	6
pi	0.67	0.33	
-pi*log(pi)	0.39	0.53	0.92

Total Entropy = 0.69

CGPA = Medium

	Yes	No	Total
Count	4	3	7
pi	0.57	0.43	
-pi*log(pi)	0.46	0.52	0.99

Information Gain = 0.30

CGPA = Low

	Yes	No	Total
Count	0	5	5
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

(c) Splitted data set (based on the feature 'Communication'):

Communication = Good

	Yes	No	Total
Count	7	2	9
pi	0.78	0.22	
-pi*log(pi)	0.28	0.48	0.76

Total Entropy = 0.63

Communication = Bad

	Yes	No	Total
Count	1	8	9
pi	0.11	0.89	
-pi*log(pi)	0.35	0.15	0.50

Information Gain = 0.36

(d) Splitted data set (based on the feature 'Aptitude'):

Aptitude = High

	Yes	No	Total
Count	8	3	11
pi	0.73	0.27	
-pi*log(pi)	0.33	0.51	0.85

Total Entropy = 0.52

Aptitude = Low

	Yes	No	Total
Count	0	7	7
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

Information Gain = 0.47

(e) Splitted data set (based on the feature 'Programming Skill'):

Programming Skill = Good

	Yes	No	Total
Count	5	4	9
pi	0.56	0.44	
-pi*log(pi)	0.47	0.52	0.99

Total Entropy = 0.95

Programming Skill = Bad

	Yes	No	Total
Count	3	6	9
pi	0.33	0.67	
-pi*log(pi)	0.53	0.39	092

Information Gain = 0.04

Decision Tree Implementation...

- among all the features, 'Aptitude' results in the best information gain when adopted for the split.
- So, at the first level, 'Aptitude' will be the first node of the decision tree.
- Because, Aptitude = Low, Entropy is 0, which indicates that always the result will be the same irrespective of the values of the other features.
- Hence, the branch towards Aptitude = Low will not continue any further.

(a) Original data set:

	Yes	No	Total
Count	8	10	18
pi	0.44	0.56	
-pi*log(pi)	0.52	0.47	0.99

Total Entropy = 0.99

(b) Splitted data set (based on the feature 'CGPA'):

CGPA = High

	Yes	No	Total
Count	4	2	6
pi	0.67	0.33	
-pi*log(pi)	0.39	0.53	0.92

Total Entropy = 0.69

CGPA = Medium

	Yes	No	Total
Count	4	3	7
pi	0.57	0.43	
-pi*log(pi)	0.46	0.52	0.99

Information Gain = 0.30

CGPA = Low

	Yes	No	Total
Count	0	5	5
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

(c) Splitted data set (based on the feature 'Communication'):

Communication = Good

	Yes	No	Total
Count	7	2	9
pi	0.78	0.22	
-pi*log(pi)	0.28	0.48	0.76

Total Entropy = 0.63

Communication = Bad

	Yes	No	Total
Count	1	8	9
pi	0.11	0.89	
-pi*log(pi)	0.35	0.15	0.50

Information Gain = 0.36

(d) Splitted data set (based on the feature 'Aptitude'):

Aptitude = High

	Yes	No	Total
Count	8	3	11
pi	0.73	0.27	
-pi*log(pi)	0.33	0.51	0.85

Total Entropy = 0.52

Aptitude = Low

	Yes	No	Total
Count	0	7	7
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

Information Gain = 0.47

(e) Splitted data set (based on the feature 'Programming Skill'):

Programming Skill = Good

	Yes	No	Total
Count	5	4	9
pi	0.56	0.44	
-pi*log(pi)	0.47	0.52	0.99

Total Entropy = 0.95

Programming Skill = Bad

	Yes	No	Total
Count	3	6	9
pi	0.33	0.67	
-pi*log(pi)	0.53	0.39	092

Information Gain = 0.04

- As a part of level 2, we will thus have only one branch to navigate in this case – the one for Aptitude = High.
- the entropy value
- 0.85 before the split
- 0.33 when the feature 'CGPA' is used for split
- 0.30 when the feature 'Communication' is used for split
- 0.80 when the feature 'Programming skill' is used for split

(a) Level 2 starting set:

	Yes	No	Total
Count	8	3	11
pi	0.73	0.27	
-pi*log(pi)	0.33	0.51	0.85

Total Entropy = 0.85

(b) Splitted data set (based on the feature 'CGPA'):

CGPA = High

	Yes	No	Total
Count	4	0	4
pi	1.00	0.00	
-pi*log(pi)	0.00	0.00	0.00

CGPA = Medium

	Yes	No	Total
Count	4	1	5
pi	0.80	0.20	
-pi*log(pi)	0.26	0.46	0.72

CGPA = Low

	Yes	No	Total
Count	0	2	2
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

Total Entropy = 0.33

Information Gain = 0.52

(c) Splitted data set (based on the feature 'Communication'):

Communication = Good

	Yes	No	Total
Count	7	0	7
pi	1.00	0.00	
-pi*log(pi)	0.00	0.00	0.00

Total Entropy = 0.30

Communication = Bad

	Yes	No	Total
Count	1	3	4
pi	0.25	0.75	
-pi*log(pi)	0.50	0.31	0.81

Information Gain = 0.55

(d) Splitted data set (based on the feature 'Programming Skill'):

Programming Skill = Good

	Yes	No	Total
Count	5	1	6
pi	0.83	0.17	
-pi*log(pi)	0.22	0.43	0.65

Total Entropy = 0.80

Programming Skill = Bad

	Yes	No	Total
Count	3	2	5
pi	0.60	0.40	
-pi*log(pi)	0.44	0.53	0.97

Information Gain = 0.05

Decision Tree Implementation...

- As a part of level 3, we will thus have only one branch to navigate in this case – the one for Communication = Bad.
- As can be seen from the figure, the entropy value is as follows:
- 0.81 before the split
- 0 when the feature 'CGPA' is used for split
- 0.50 when the feature 'Programming Skill' is used for split

Aptitude = High & Communication = Bad

CGPA	Programming Skill	Job offered?
High	Good	Yes
Low	Bad	No
Low	Bad	No
Medium	Good	No

(a) Level 2 starting set:

	Yes	No	Total
Count	1	3	4
pi	0.25	0.75	
-pi*log(pi)	0.50	0.31	0.81

Total Entropy = 0.81

(b) Splitted data set (based on the feature 'CGPA'):

CGPA = High

	Yes	No	Total
Count	1	0	1
pi	1.00	0.00	
-pi*log(pi)	0.00	0.00	0.00

Total Entropy = 0.00

CGPA = Medium

	Yes	No	Total
Count	0	1	1
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

CGPA = Low

	Yes	No	Total
Count	0	2	2
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

Information Gain = 0.81

(c) Splitted data set (based on the feature 'Programming Skill'):

Programming Skill = Good

	Yes	No	Total
Count	1	1	2
pi	0.50	0.50	
-pi*log(pi)	0.50	0.50	1.00

Total Entropy = 0.50

Programming Skill = Bad

	Yes	No	Total
Count	0	2	2
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

Information Gain = 0.31

- Hence, the information gain after split with the feature CGPA is 0.81, which is the maximum possible information gain
- Then the tree will not continue any further.

Aptitude = High & Communication = Bad

CGPA	Programming Skill	Job offered?
High	Good	Yes
Low	Bad	No
Low	Bad	No
Medium	Good	No

(a) Level 2 starting set:

	Yes	No	Total
Count	1	3	4
pi	0.25	0.75	
-pi*log(pi)	0.50	0.31	0.81

Total Entropy = 0.81

(b) Splitted data set (based on the feature 'CGPA'):

CGPA = High

	Yes	No	Total
Count	1	0	1
pi	1.00	0.00	
-pi*log(pi)	0.00	0.00	0.00

Total Entropy = 0.00

CGPA = Medium

	Yes	No	Total
Count	0	1	1
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

Information Gain = 0.81

CGPA = Low

	Yes	No	Total
Count	0	2	2
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

Machine Learning
Subject Code: 20A05602T

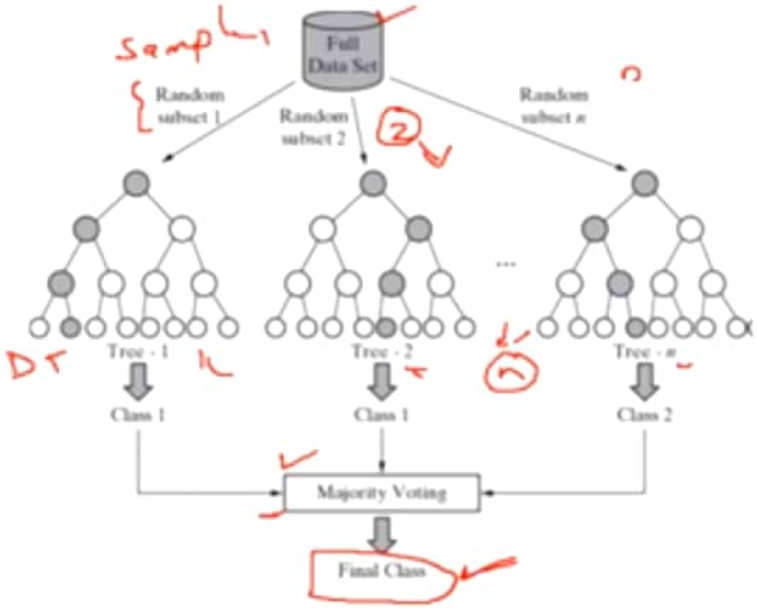
UNIT III - Supervised Learning: Classification - 3-2-5

Random Forest Algorithm

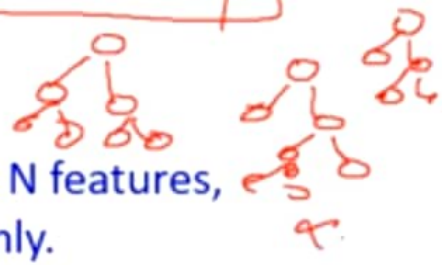
- What is Random Forest Model
- Algorithm
- Problems in Random Forest
- Out-of-bag (OOB) error in random forest
- Strengths and weakness
- Application of random forest

Random Forest Model

- Random forest is an ensemble classifier.
- A combining classifier that uses and combines many decision tree classifiers.
- Ensembling is usually done using the concept of bagging with different feature sets.
- Large number of trees used in random forest, and train the trees,
- Then the random forest is generated by combining these trees,
- The majority vote is applied to combine the output of the different trees.
- The result of ensemble model is usually better than that from the individual decision tree models.



The Random Forest Algorithm



- The random forest algorithm works as follows:
- 1. If there are N variables or features in the input data set,
- select a subset of 'm' (m < N) features at random out of the N features,
- the observations or data instances should be picked randomly.
- 2. Use the best split principle on these 'm' features to calculate the number of nodes 'd'.
- 3. Keep splitting the nodes to child nodes till the tree is grown to the maximum possible extent.
- 4. Select a different subset of the training data 'with replacement' to train another decision tree following steps (1) to (3).
- Repeat this to build and train 'n' decision trees.
- 5. Final class assignment is done on the basis of the majority votes from the 'n' trees.

Problems in Random Forest

- In the random forest classifier, if the number of trees is assumed to be excessively large, the model may get overfitted.
- In an extreme case of overfitting, the model may mimic the training data, and training error might be almost 0.
- When the model is run on an unseen sample, it may result in a very high validation error.

Out-Of-Bag (OOB) Error in Random Forest

- In random forests, each tree is constructed using a different bootstrap sample from the original data.
- The samples left out of the bootstrap and not used in the construction of the i-th tree can be used to measure the performance of the model.
- At the end of the run, predictions for each such sample evaluated each time are tallied, and the final prediction for that sample is obtained by majority voting.
- The total error rate of predictions for such samples is termed as **out-of-bag (OOB) error rate.**
- The OOB error rate shown in the confusion matrix.
- Because of this reason, the error rate displayed is often surprisingly high.

Strengths of random forest

- It runs efficiently on large and expansive data sets.
- It is a best method to work in large number of errored or missing data in data set.
- It has powerful techniques for balancing errors in a class population of unbalanced data sets.
- It gives estimates (or assessments) about which features are the most important ones in the overall classification.
- It generates an internal unbiased estimate of the generalization error as the forest generation progresses.
- Generated forests can be saved for future use on other data.
- The random forest algorithm can be used to solve both classification and regression problems.

Weaknesses of random forest

- The random forest, combines a number of decision tree models, is difficult to understand as a decision tree model.
- It is computationally much more expensive than a simple model like decision tree.

Application of random forest

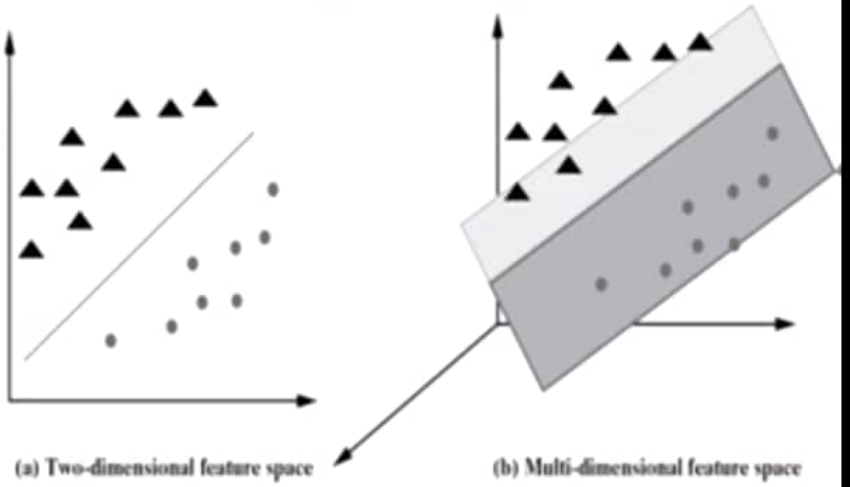
- Banking Industry
  - Credit Card Fraud Detection
  - Customer Segmentation
  - Predicting Loan Defaults
- Healthcare and Medicine
  - Cardiovascular Disease Prediction
  - Diabetes Prediction
  - Breast Cancer Prediction
- Stock Market
  - Stock Market Prediction
  - Stock Market Sentiment Analysis
  - Bitcoin Price Detection
- E-Commerce
  - Product Recommendation
  - Price Optimization
  - Search Ranking

Machine Learning
Subject Code: 20A05602T

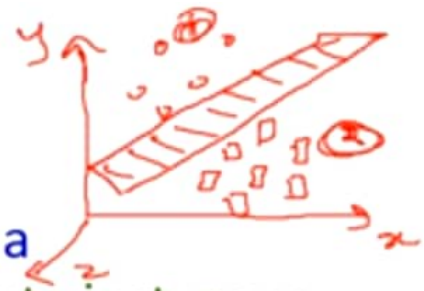
UNIT III - Supervised Learning: Classification - 3-2-6

Support Vector Machines (SVM)

- Classification using hyperplanes
- Identifying the correct hyperplane
- Maximum margin hyperplane
- Kernel trick
- Strengths and Weaknesses
- Applications



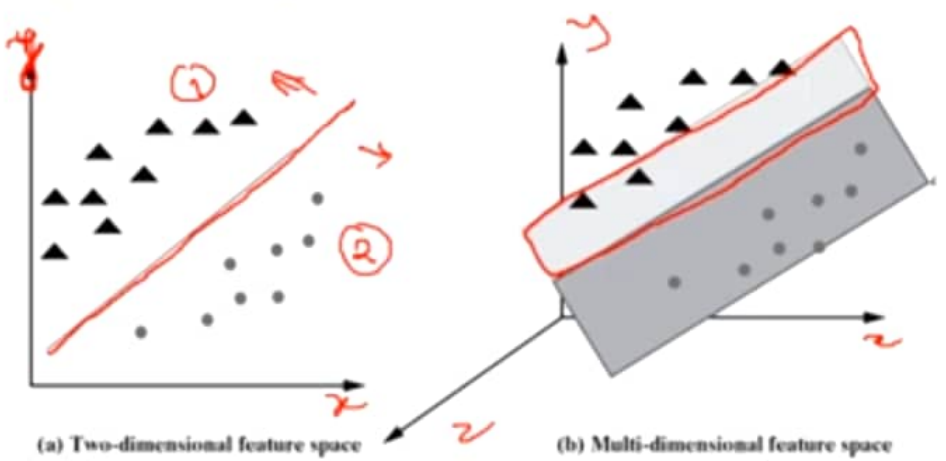
Support Vector Machine - SVM



- SVM is based on the concept of a surface, called a hyperplane, which draws a boundary between data instances plotted in the multi-dimensional feature space.
- The output prediction of an SVM is one of two conceivable classes which are already defined in the training data.
- **The SVM algorithm builds an N-dimensional hyperplane model that assigns future instances into one of the two possible output classes.**
- SVM supports linear classification and regression.

Classification using hyperplanes

- The SVM model used to classify the data belonging to different classes.
- Simple Example - the data instances triangles and circles are linearly separable.
- when mapped in a two dimensional space, the data instances belonging to different classes fall in different sides of a straight line drawn in the two-dimensional space as depicted in Figure a.
- If the same concept is extended to a multidimensional feature space, the straight line dividing data instances belonging to different classes transforms to a hyperplane as depicted in Figure b.



(a) Two-dimensional feature space

(b) Multi-dimensional feature space

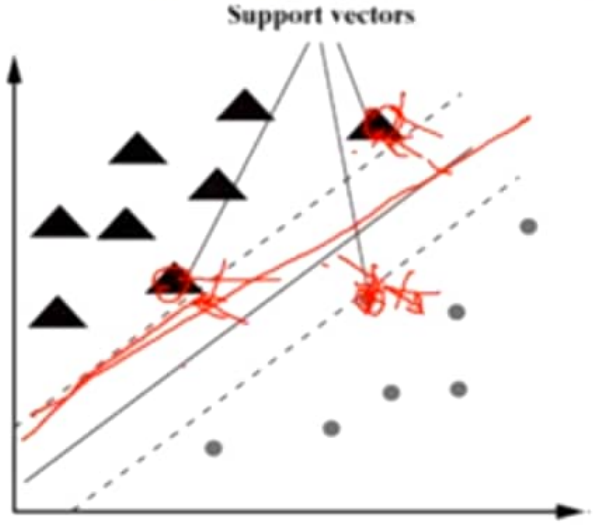
Classification using hyperplanes...

- Training data sets which have a significant grouping margin, will function well with SVM.
- The important terms used in SVM are
  - generalization error,
  - support vector,
  - hyperplane and
  - margin.

- Generalization error in SVM is the measure of how accurately this SVM model can predict new data.
- SVM model is inflexible in classification and tries to fit the hyperplane in the training set, thereby causing overfitting.

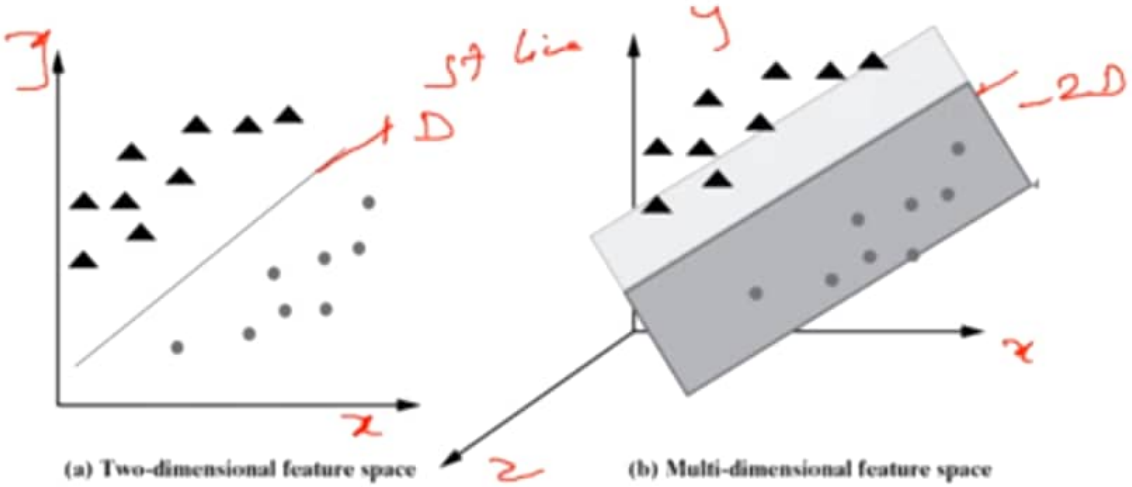
Support Vectors:

- Support vectors are the data points (representing classes), the critical component in a data set, which are near the identified set of lines (hyperplane).
- If support vectors are removed, then alter the position of the dividing hyperplane.



Hyperplane:

- For an N-dimensional feature space, hyperplane is a flat subspace of dimension (N−1) that separates and classifies a set of data.
- it is difficult to visualize a feature space greater than three dimensions, much like for a subspace or hyperplane having more than three dimensions.



(a) Two-dimensional feature space (b) Multi-dimensional feature space

Hyperplane...

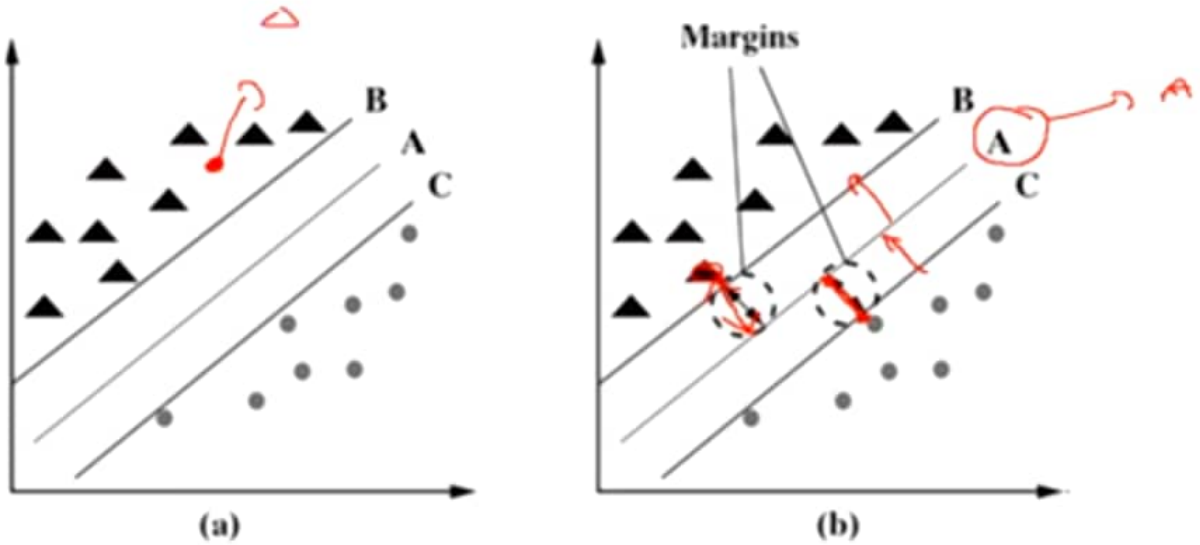
2-1 - 1D

- Mathematically, in a two-dimensional space, hyperplane can be defined by the equation:
- c0 + c1 X1 + c2 X2 = 0, an equation of a straight line.
- Extending this concept to an N-dimensional space, hyperplane can be defined by the equation: N-1 D
- c0 + c 1X1 + c2 X2 + ... + cN XN = 0 can be represented as follows:

$$\vec{c}.\vec{X} + c_0 = 0$$

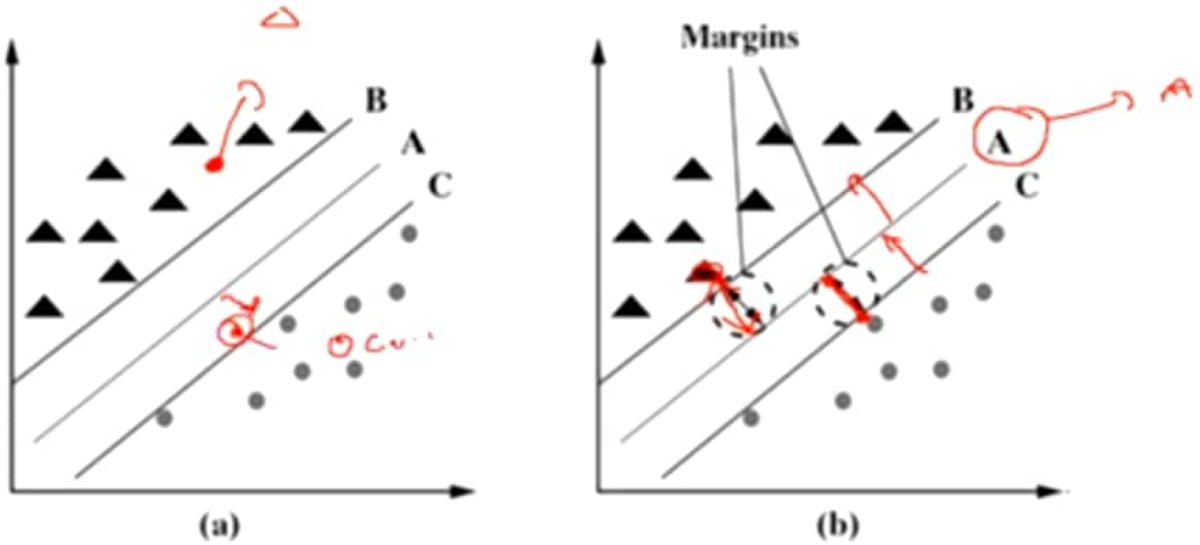
Margin:

- The distance between hyperplane and data points is known as margin.
- when a new testing data point/data set is added, the side of the hyperplane decide the class.



Margin:

- The distance between hyperplane and data points is known as margin.
- when a new testing data point/data set is added, the side of the hyperplane decide the class.

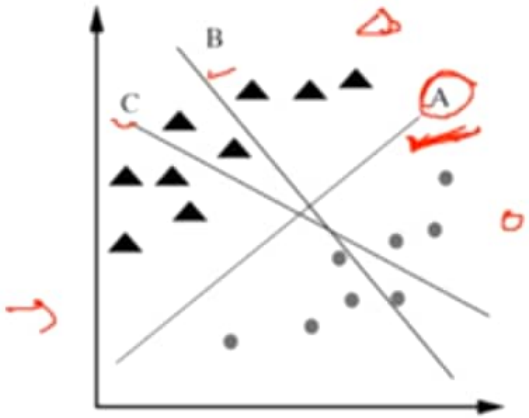


Identifying the correct hyperplane in SVM

- there are multiple options for hyperplanes dividing the data instances belonging to the different classes.
- We need to identify which one will result in the best classification.
- Let us examine a few scenarios, and simplicity of visualization, the hyperplanes have been shown as straight lines in most of the diagrams.

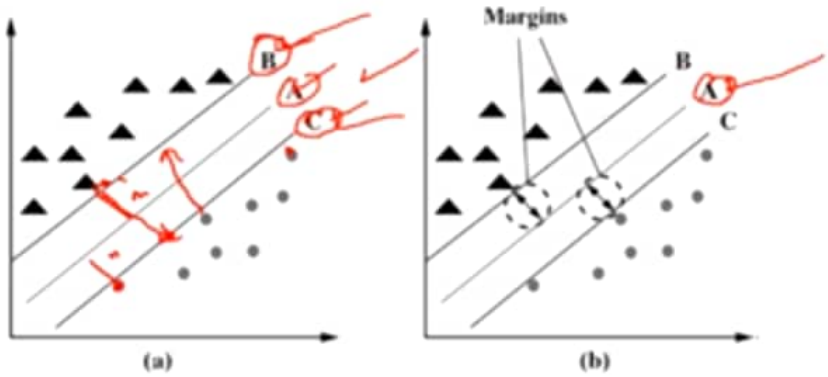
Scenario 1

- two classes of data points triangles and circles.
- we have three hyperplanes: A, B, and C.
- identify the correct hyperplane which better segregates the two classes represented by the triangles and circles.
- hyperplane 'A' has performed this task quite well.

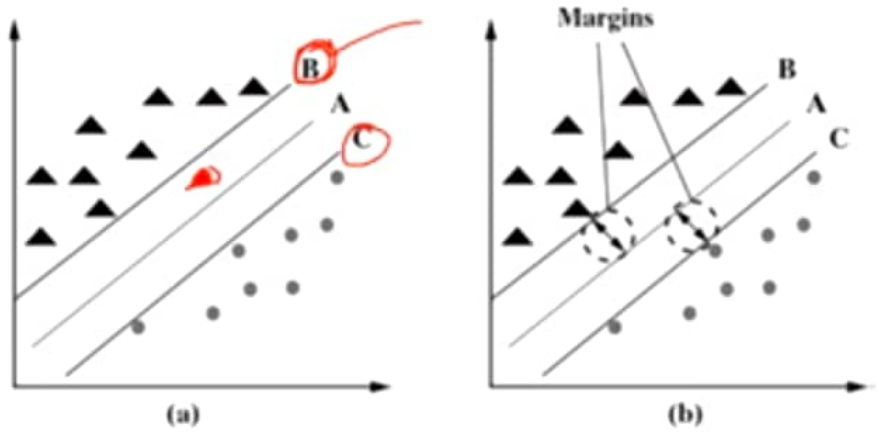


Scenario 2

- we have three hyperplanes: A, B, and C.
- identify the correct hyperplane which classifies the triangles and circles in the best possible way.
- Here, maximizing the distances between the nearest data points of both the classes and hyperplane will help us decide the correct hyperplane.
- This distance is called as margin.

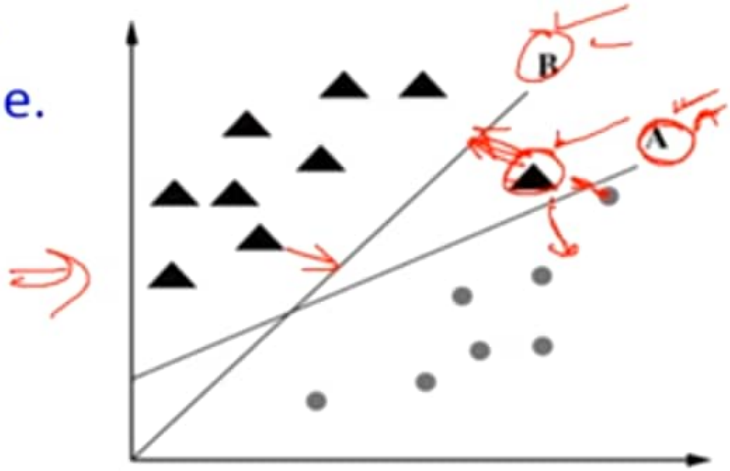


- the margin for hyperplane A is high as compared to those for both B and C.
- hyperplane A is the correct hyperplane.
- If we select a hyperplane having a lower margin (distance), then there is a high probability of misclassification.



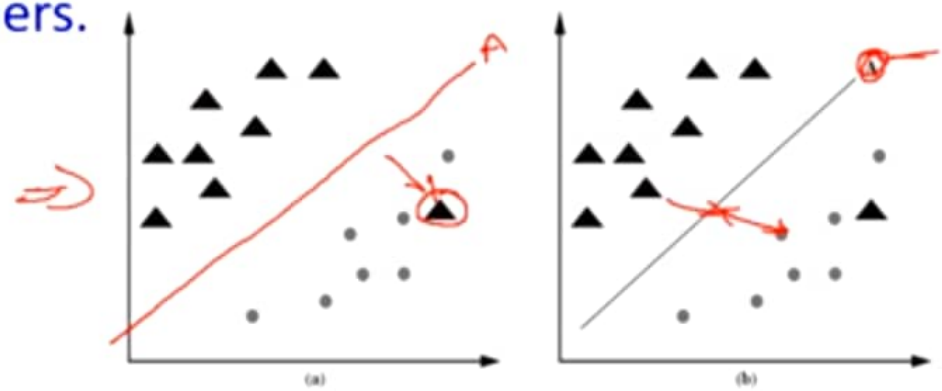
Scenario 3

- hyperplane B as it has a higher margin (distance from the class) than A.
- SVM selects the hyperplane which classifies the classes accurately before maximizing the margin.
- Here, hyperplane B has a classification error, and A has classified all data instances correctly.
- Therefore, A is the correct hyperplane.



Scenario 4

- it is not possible to segregate the two classes by using a straight line, as one data instance belonging other class (circle) as an outlier.
- SVM has a feature to ignore outliers and find the hyperplane that has the maximum margin.
- hyperplane A has divides the class with maximum margin, hence, SVM is robust to outliers.



Characteristics of SVM

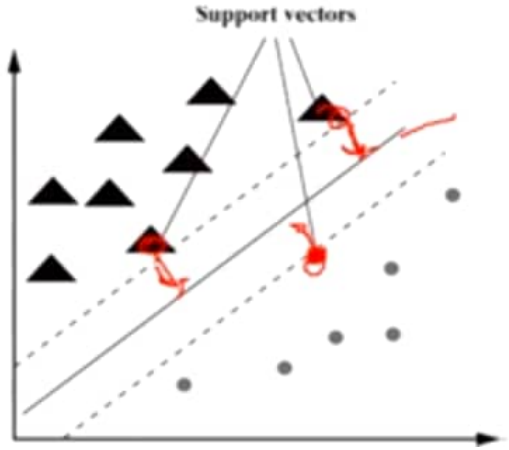
- So, by summarizing the observations from the different scenarios, we can say that
  1. The hyperplane should segregate the data instances belonging to the two classes in the best possible way.
  2. It should maximize the distances between the nearest data points of both the classes, i.e. maximize the margin.
  3. If there is a need to prioritize between higher margin and lesser misclassification, the hyperplane should try to reduce misclassifications.

Maximum Margin Hyperplane (MMH)

- Finding the Maximum Margin Hyperplane (MMH) is, identifying the hyperplane which has the largest separation with the data instances of the two classes.
- This helps to achieve more generalization and hence less number of issues in the classification of unknown data.

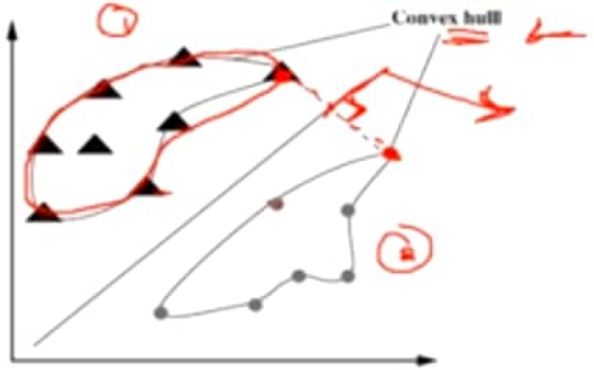
MMH - Support vectors

- Support vectors, are data instances from the two classes which are closest to the MMH.
- There should be at least one support vector from each class.
- modelling a problem using SVM is identifying the support vectors and MMH corresponding to the problem space.



Identifying the MMH for linearly separable data

- Finding out the MMH is relatively straightforward for the data that is linearly separable.
- an outer boundary needs to be drawn for the data instances belonging to the different classes, these outer boundaries are known as convex hull.
- The MMH can be drawn as the perpendicular bisector of the shortest line (i.e. the connecting line having the shortest length) between the convex hulls.



Hyperplane in N-dimensional feature

- a hyperplane in the N-dimensional feature space can be represented by the equation: $\vec{c}.\vec{X} + c_0 = 0$
- Using this equation, the objective is to find a set of values for the vector $\vec{c}$ such that two hyperplanes, represented by the equations below, can be specified

$$\vec{c}.\vec{X} + c_0 \geq +1$$

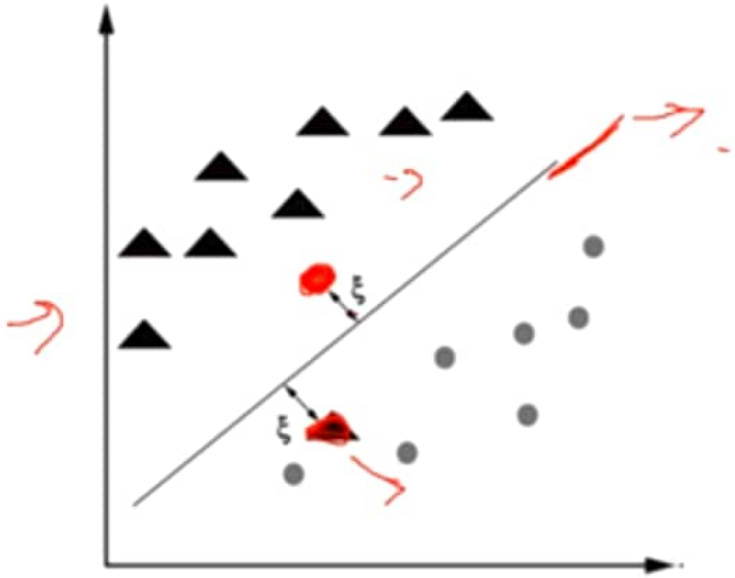
$$\vec{c}.\vec{X} + c_0 \leq -1$$

Hyperplane in N-dimensional feature...

- This is to ensure that
  - all the data instances that belong to one class falls above one hyperplane and
  - all the data instances belonging to the other class falls below another hyperplane.
- According to vector geometry, the distance of these planes should be $\frac{2}{\vec{c}}$
- In order to maximize the distance between hyperplanes, the value of $\vec{c}$ should be minimized.
- the task of SVM is to solve the optimization problem: $\min\left(\frac{1}{2}\vec{c}\right)$

Identifying the MMH for non-linearly separable data

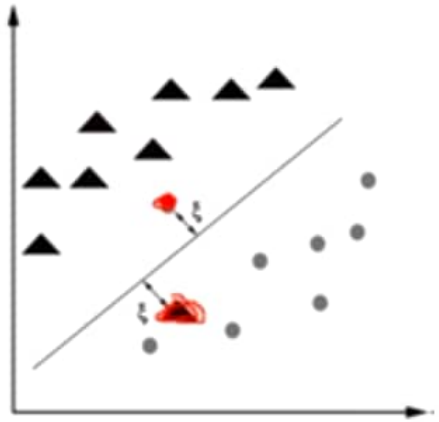
- A slack variable ξ, which provides some soft margin for data instances in one class that fall on the wrong side of the hyperplane.



Identifying the MMH for non-linearly separable data...

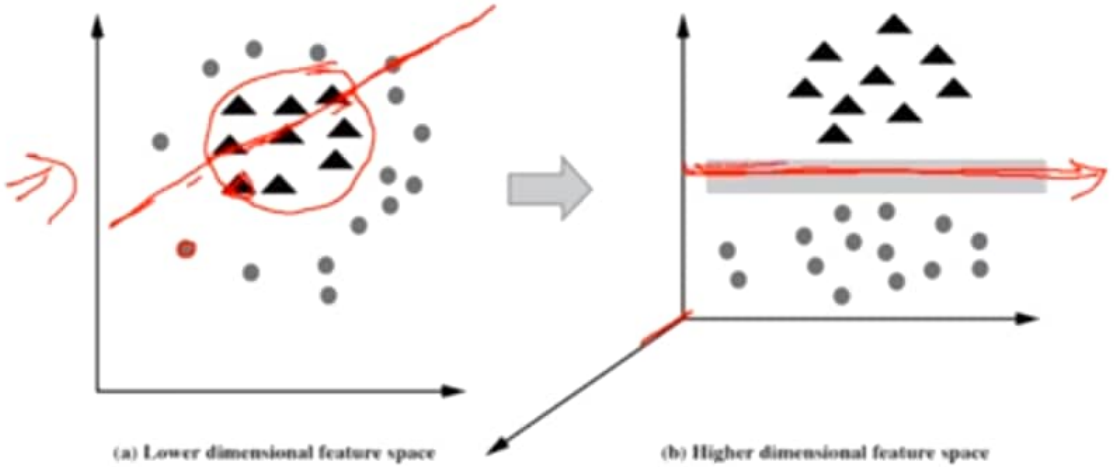
- A cost value '**C**' is computed on all such data instances that fall on the wrong side of the hyperplane.
- SVM is now to minimize the total cost, due to such data instances in order to solve the revised optimization problem:

$$= \min\left(\frac{1}{2}\vec{c}^2\right) + C\sum_{i=1}^{N}\xi_i$$



Kernel Trick

- SVM has kernel trick to deal with non-linearly separable data.
- The kernel functions can transform lower dimensional input space to a higher dimensional space.
- it converts linearly non-separable data to a linearly separable data.



(a) Lower dimensional feature space

(b) Higher dimensional feature space

Kernel trick...

- Some of the common kernel functions for transforming from a lower dimension 'i' to a higher dimension 'j' used by different SVM implementations are as follows:

Linear kernel: It is in the form $K(\vec{x_i}, \vec{x_j}) = \vec{x_i}.\vec{x_j}$

Polynomial kernel: It is in the form $K(\vec{x_i}, \vec{x_j}) = (\vec{x_i}.\vec{x_j} + 1)^d$

Sigmoid kernel: It is in the form $K(\vec{x_i}, \vec{x_j}) = \tan h(k\vec{x_i}.\vec{x_j} - \partial)$

Gaussian RBF kernel: It is in the form $K(\vec{x_i}, \vec{x_j}) = e^{\frac{-\vec{x_i} - \vec{x_j}^2}{2\sigma^2}}$

Kernel trick...

- When data instances of the classes are closer to each other, this method can be used.
- The effectiveness of SVM depends both on the
  - selection of the kernel function
  - adoption of values for the kernel parameters

Strengths and Weaknesses of SVM

Strengths of SVM

- SVM can be used for both classification and regression.
- It is robust, i.e. not much impacted by data with noise or outliers.
- The prediction results using this model are very promising.

Weaknesses of SVM

- SVM is applicable only for binary classification, i.e. when there are only two classes in the problem domain.
- The SVM model is very complex – almost like a black box when it deals with a high-dimensional data set.
- Hence, it is very difficult to understand the model in such cases.
- It is slow for a large dataset, i.e. a data set with either a large number of features or a large number of instances.
- It is quite memory-intensive.

Application of SVM

- Face detection – SVMc classify parts of the image as a face and non-face and create a square boundary around the face.
- Text and hypertext categorization – SVM use training data to classify documents into different categories. It categorizes on the basis of the score generated and then compares with the threshold value.
- Classification of images – It provides better accuracy in comparison to the traditional query-based searching techniques.
- Bioinformatics – It includes protein classification and cancer classification. We use SVM for identifying the classification of genes, patients on the basis of genes and other biological problems.
- Handwriting recognition – to recognize handwritten characters used widely.