UNIT III TREE AND PROBABILISTIC MODELS

LEARNING WITH TREES

- A decision tree is a powerful data structure which implements divide-and-conquer strategy.
- The computational cost of making the tree is fairly low, but the cost of using it is even lower: O(log N), where N is the number of data points.
- This is sufficient to make trees seem attractive for machine learning.
- It is an efficient nonparametric method, which can be used for both classification and regression.(CART)

Parametric estimation:

- Define a model over the whole input space and learn its parameters from all of the training data. Then use the same model and the same parameter set for any test input.

Non- Parametric estimation

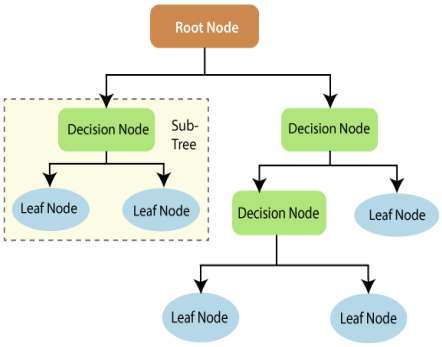
- Input space is divided into local regions, defined by a distance measure (Euclidean norm) and for each input, the corresponding local model computed from the training data in that region is used.
- A decision tree is a hierarchical model for supervised learning whereby the local region is identified in a sequence of recursive splits in a smaller number of steps.

DECISION TREE

- A decision tree is composed of internal decision nodes and terminal leaves.
- Each decision node m implements a test function fm(x) with discrete outcomes labeling the branches.
- Given an input, at each node, a test is applied and one of the branches is taken depending on the outcome.

- This process starts at the root and is repeated recursively until a leaf node is hit, at which point the value written in the leaf constitutes the output.

Decision Tree



Types of Decisions

There are two main types of decision trees that are based on the target variable, i.e., categorical variable decision trees and continuous variable decision trees.

1. Categorical variable decision tree

A categorical variable decision tree includes categorical target variables that are divided into categories. For example, the categories can be yes or no. The categories mean that every stage of the decision process falls into one of the categories, and there are no in-betweens.

2. Continuous variable decision tree

A continuous variable decision tree is a decision tree with a continuous target variable. For example, the income of an individual whose income is unknown can be predicted based on available information such as their occupation, age, and other continuous variables.

CONSTRUCTING DECISION TREES

When to Consider Decision Trees

- Instances are represented by attribute-value pairs.
  - Fixed set of attributes, and the attributes take a small number of disjoint possible values.

- The target function has discrete output values.

  – Decision tree learning is appropriate for a boolean classification, but it easily extends to learning functions with more than two possible output values.
- Disjunctive descriptions may be required.

  – decision trees naturally represent disjunctive expressions.

The training data may contain errors.

– Decision tree learning methods are robust to errors, both errors in classifications of the training

examples and errors in the attribute values that describe these examples.

• The training data may contain missing attribute values.

– Decision tree methods can be used even when some training examples have unknown values.

• Decision tree learning has been applied to problems such as learning to classify

– medical patients by their disease,

– equipment malfunctions by their cause, and

– loan applicants by their likelihood of defaulting on payments.

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.

In Decision Tree the major challenge is to identification of the attribute for the root node in each level. This process is known as attribute selection. We have two popular attribute selection measures:

1. Information Gain
2. Gini Index

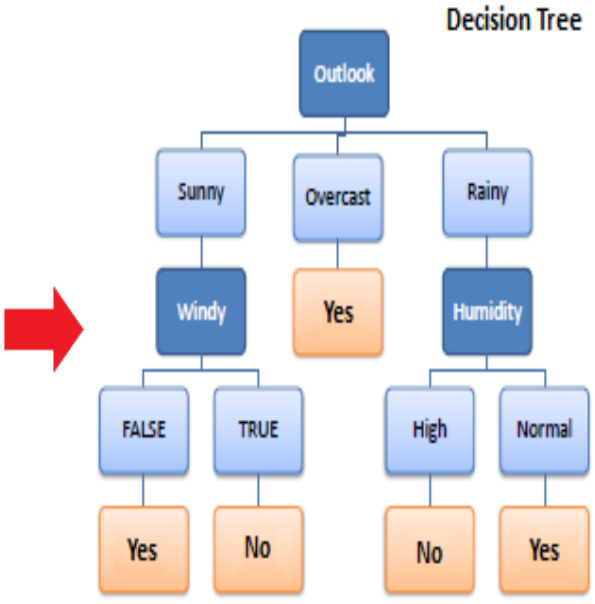
The core algorithm for building decision trees called ID3 by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses Entropy and Information Gain to construct a decision tree. Decision tree includes all predictors with the dependence assumptions between predictors.

• Entropy

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.

Predictors | Target

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No



To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:

a) Entropy using the frequency table of one attribute:

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

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

b) Entropy using the frequency table of two attributes:

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)
= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971
= 0.693

Information Gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

Step 1: Calculate entropy of the target.

$$\text{Entropy(PlayGolf)} = \text{Entropy}(5,9)$$
$$= \text{Entropy}(0.36, 0.64)$$
$$= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64)$$
$$= 0.94$$

Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

$$G(\text{PlayGolf, Outlook}) = E(\text{PlayGolf}) - E(\text{PlayGolf, Outlook})$$
$$= 0.940 - 0.693 = 0.247$$

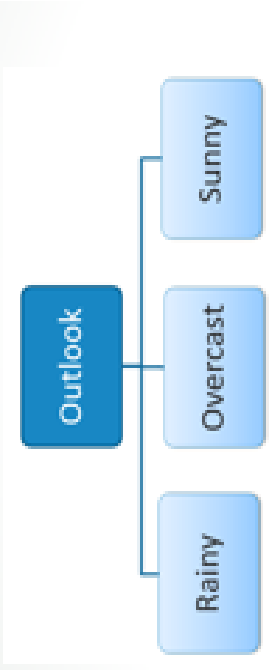
		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

Step 3: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

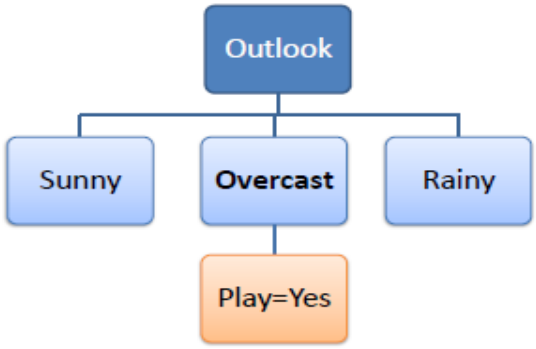


Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

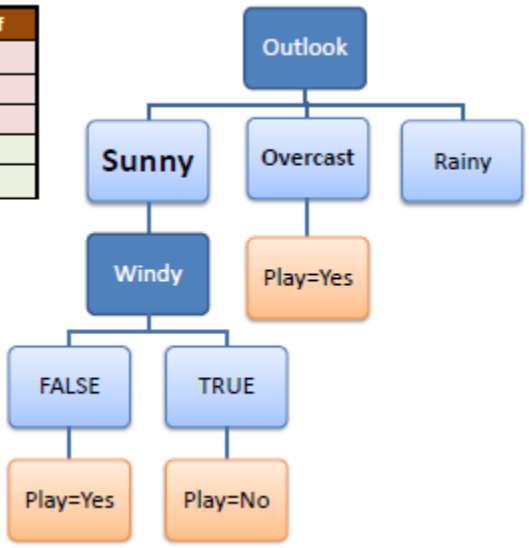
Step 4a: A branch with entropy of 0 is a leaf node.

Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



Step 4b: A branch with entropy more than 0 needs further splitting.

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

Decision Tree to Decision Rules

A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

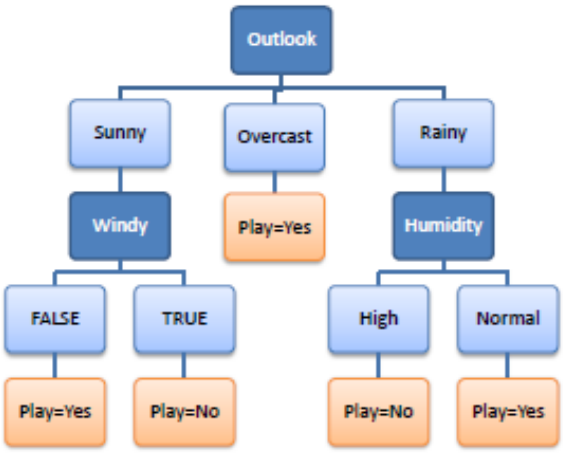
R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

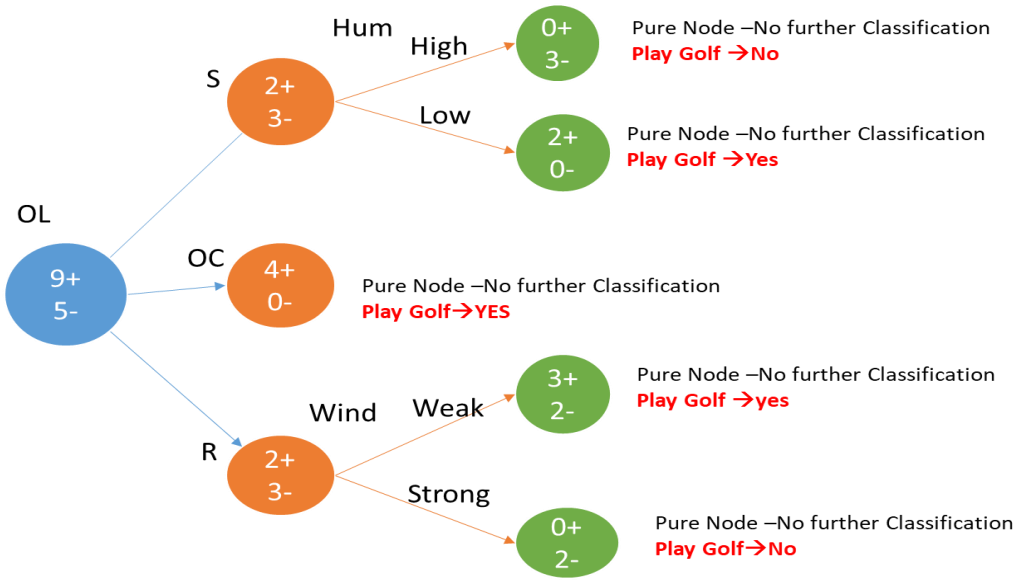
R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes



CONSTRUCTED DECISION TREE



At the root node, the information gains are:

IG(S, wind) = 0.048 IG(S, outlook) = 0.246

IG(S, humidity) = 0.151 IG(S,temperature) = 0.029

"outlook" has the maximum IG =⇒ chosen as the root node

When to Stop expanding a node further:

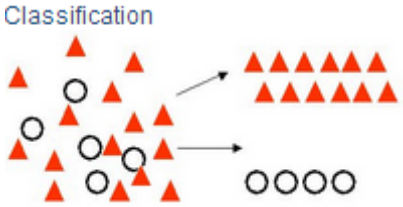
- It consist of examples all having the same label (the node becomes "pure")
- Or we run out of features to test!

The ID3 Algorithm

- If all examples have the same label:
  - return a leaf with that label
- Else if there are no features left to test:
  - return a leaf with the most common label
- Else:
  - choose the feature $\hat{F}$ that maximises the information gain of S to be the next node using Equation (12.2)
  - add a branch from the node for each possible value f in $\hat{F}$
  - for each branch:
    - calculate S_f by removing $\hat{F}$ from the set of features
    - recursively call the algorithm with S_f, to compute the gain relative to the current set of examples

CLASSIFICATION AND REGRESSION TREES

- There is another well-known tree-based algorithm, CART
- It can be used for both classification and regression.
- The only real difference with classification in CART is that a different information measure is commonly used. (Gini Impurity)
- Regression-> Suppose that the outputs are continuous, so that a regression model is appropriate.
- None of the node impurity measures, we'll go with sum-of-squares error.
- To evaluate the choice of which feature to use next, we also need to find the value at which to split the dataset according to that feature.
- Example
  - Classification Trees: where the target variable is categorical and the tree is used to identify the "class" within which a target variable would likely fall into.



  - Regression Trees: where the target variable is continuous and tree is used to predict it's value.

Gini index

Gini index is a metric for classification tasks in CART. It stores sum of squared probabilities of each class. We can formulate it as illustrated below.

Gini = $1 - \Sigma (Pi)^2$ for i=1 to number of classes

Let us consider the play golf example

Outlook

Outlook is a nominal feature. It can be sunny, overcast or rain. I will summarize the final decisions for outlook feature.

Outlook	Yes	No	Number of instances
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5

Gini(Outlook=Sunny) = $1 - (2/5)^2 - (3/5)^2 = 1 - 0.16 - 0.36 = 0.48$

Gini(Outlook=Overcast) = $1 - (4/4)^2 - (0/4)^2 = 0$

Gini(Outlook=Rain) = $1 - (3/5)^2 - (2/5)^2 = 1 - 0.36 - 0.16 = 0.48$

Then, we will calculate weighted sum of gini indexes for outlook feature.

Gini(Outlook) = (5/14) x 0.48 + (4/14) x 0 + (5/14) x 0.48 = 0.171 + 0 + 0.171 = 0.342

Similarly calculate the Gini index for reaming features also.

Time to decide

We've calculated gini index values for each feature. The winner will be outlook feature because its cost is the lowest.

Feature	Gini index
Outlook	0.342
Temperature	0.439
Humidity	0.367
Wind	0.428

- Sub dataset in the overcast leaf has only yes decisions. This means that overcast leaf is over.
- Apply same principles to those sub datasets in the following steps.
- Focus on the sub dataset for sunny outlook. We need to find the gini index scores for temperature, humidity and wind features respectively.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Decision for sunny outlook

We've calculated gini index scores for feature when outlook is sunny. The winner is humidity because it has the lowest value.

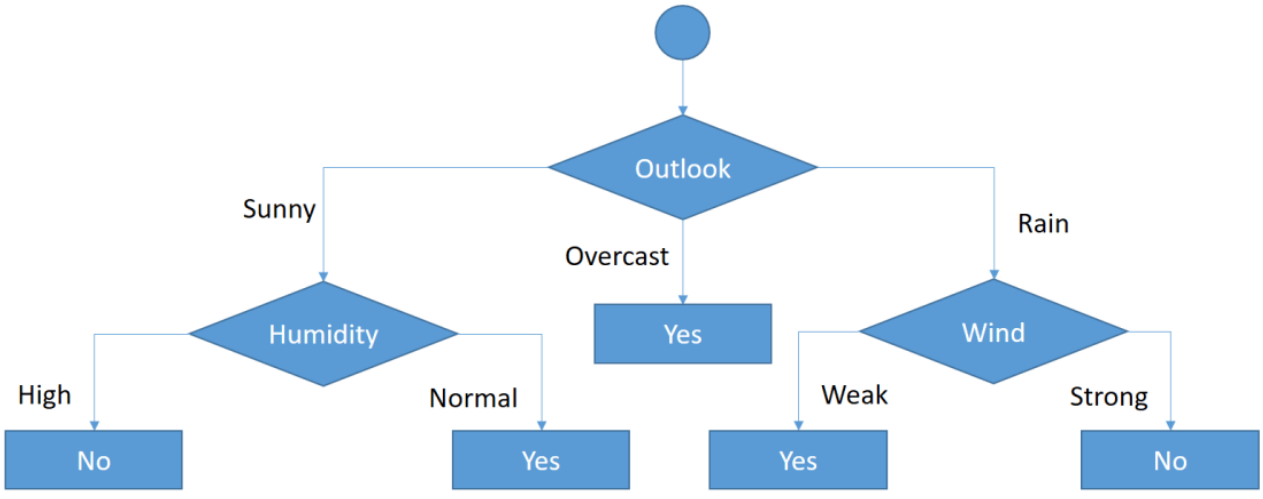
Feature	Gini index
Temperature	0.2
Humidity	0
Wind	0.466

Decision for rain outlook

The winner is wind feature for rain outlook because it has the minimum gini index score in features.

Feature	Gini index
Temperature	0.466
Humidity	0.466
Wind	0

Final decision tree built by CART



Ensemble Learning

Ensemble learning is a machine learning paradigm where multiple models (often called “weak learners”) are trained to solve the same problem and combined to get better results. The main hypothesis is that when weak models are correctly combined we can obtain more accurate and/or robust models.

Single weak learner

- In machine learning, no matter if we are facing a classification or a regression problem, the choice of the model is extremely important to have any chance to obtain good results. This choice can depend on many variables of the problem: quantity of data, dimensionality of the space, distribution hypothesis…
- A low bias and a low variance, although they most often vary in opposite directions, are the two most fundamental features expected for a model. Indeed, to be able to “solve” a problem, we want our model to have enough degrees of freedom to resolve the underlying complexity of the data we are working with, but we also want it to have not too much degrees of freedom to avoid high variance and be more robust. This is the well-known bias-variance tradeoff.

In ensemble learning theory, we call weak learners (or base models) models that can be used as building blocks for designing more complex models by combining several of them. Most of the time, these basics models perform not so well by themselves either because they have a high bias (low degree of freedom models, for example) or because they have too much variance to be robust (high degree of freedom models, for example). Then, the idea of ensemble methods is to try reducing bias and/or variance of such weak learners by combining several of them together in order to create a strong learner (or ensemble model) that achieves better performances.

Combine weak learners

- In order to set up an ensemble learning method, we first need to select our base models to be aggregated. Most of the time (including in the well-known bagging and boosting methods) a single base learning algorithm is used so that we have homogeneous weak learners that are trained in different ways. The ensemble model we obtain is then said to be "homogeneous". However, there also exist some methods that use different type of base learning algorithms: some heterogeneous weak learners are then combined into a "heterogeneous ensembles model".
- One important point is that our choice of weak learners should be coherent with the way we aggregate these models. If we choose base models with low bias but high variance, it should be with an aggregating method that tends to reduce variance whereas if we choose base models with low variance but high bias, it should be with an aggregating method that tends to reduce bias.

This brings us to the question of how to combine these models. We can mention three major kinds of meta-algorithms that aims at combining weak learners:

Bagging, that often considers homogeneous weak learners, learns them independently from each other in parallel and combines them following some kind of deterministic averaging process

Boosting, that often considers homogeneous weak learners, learns them sequentially in a very adaptive way (a base model depends on the previous ones) and combines them following a deterministic strategy

Stacking, that often considers heterogeneous weak learners, learns them in parallel and combines them by training a meta-model to output a prediction based on the different weak models predictions.

BAGGING

- The simplest method of combining classifiers is known as bagging, which stands for bootstrap aggregating, the statistical description of the method.
- A bootstrap sample is a sample taken from the original dataset with replacement, so that we may get some data several times and others not at all.
- M1, M2, M3…. … Mk

More different these models are, the better you can combine them.

Problem :-> M_i experts → get better solution

{ M1 -> Maths expert

M2 -> Logical building Expert

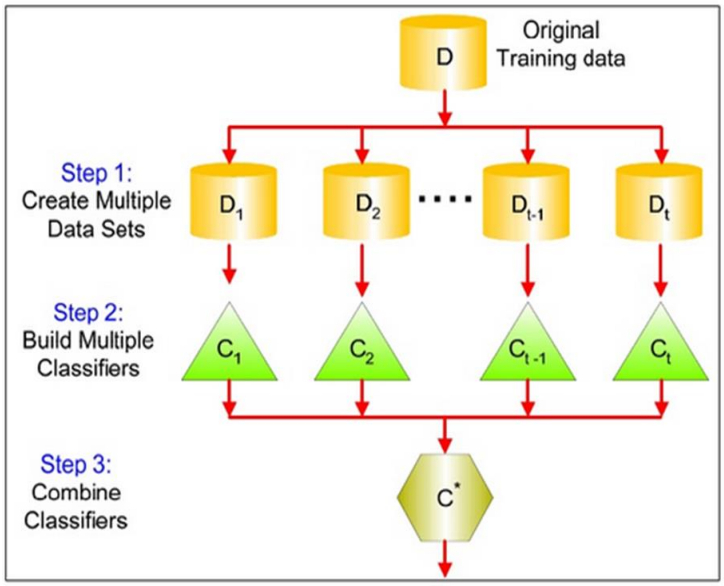
M3 -> physics Expert

:

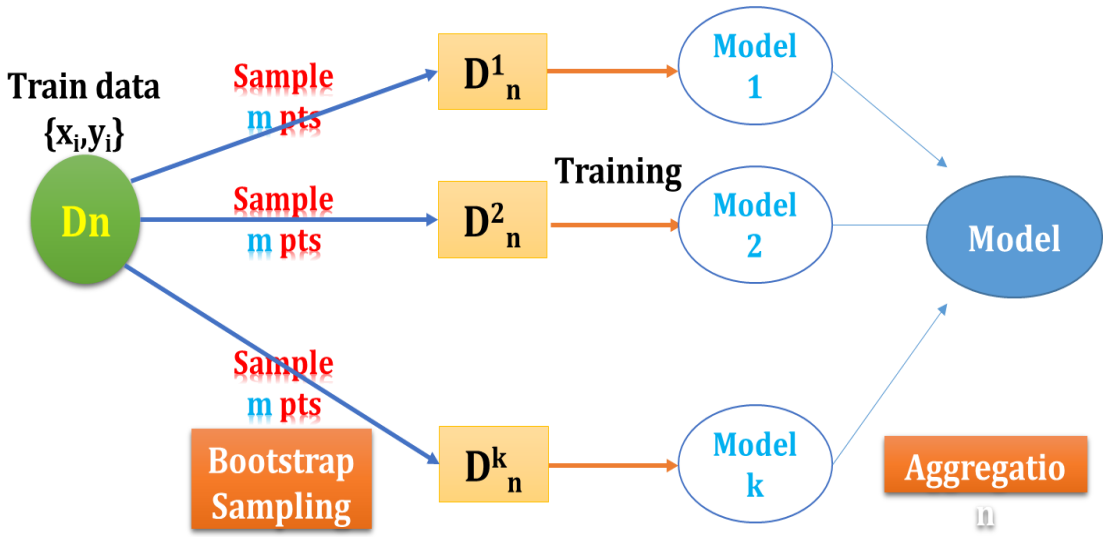
:

Mi -> some other field }

- Popular bagging Model :-Random forest



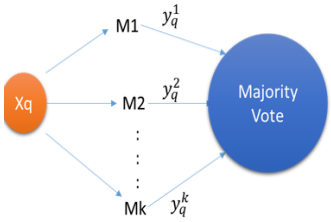
- Each Model is built with using different sample of data not using on total data.
- Each model seen different subset of data.
- Sampling with replacement – Same data points can be used for another sampling also.
- Bagging is creating Bootstrap sample→ Train the sample → Aggregate the model.



- Typical Aggregation operation in classification:
  - Majority Vote
- Typical Aggregation operation in Regression:
  - Compute Median

- Consider if "k" = 10

$\{y_q^1, y_q^2, y_q^3, \ldots\ldots y_q^k\}$



For Classification

y1	1
y2	0
y3	1
y4	0
y5	1
y6	0
y7	1
y8	1
y9	1
y10	1

Majority Vote

Yq=1

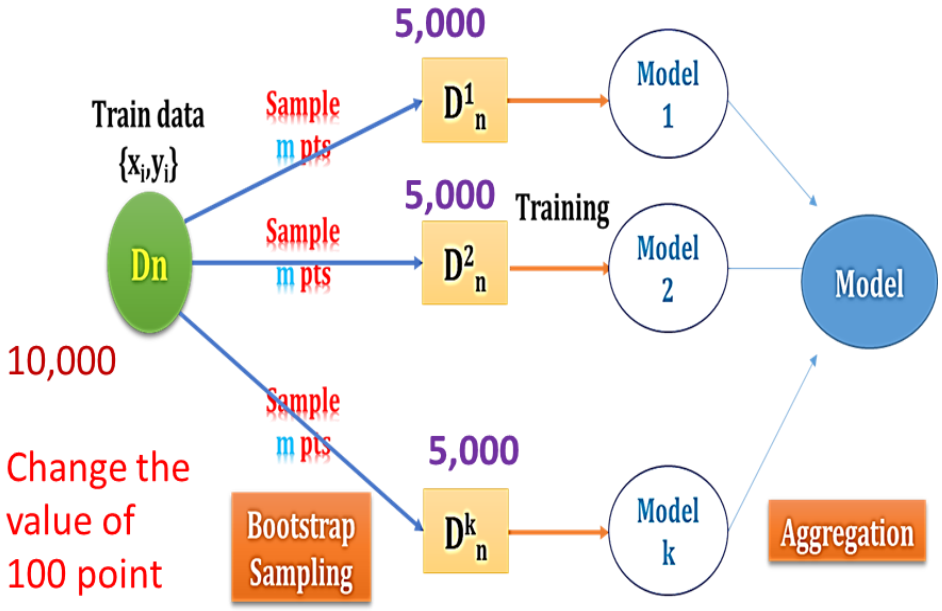
For Regression

y1	2.5
y2	2.53
y3	2.61
y4	2.7
y5	2.9
y6	2.5
y7	2.52
y8	2.53
y9	2.51
y10	2.51

Median (y_q^i)

Yq=2.52

- None of the model will see the whole data , they know about samples of data.
- Bagging can reduces the Variance in a model with out impacting the bias.
- High Variance ➔ If the model changes a lot with change in training data



- Bagging = Base Model + Sampling + Aggregation

$\{ \uparrow Variance + \downarrow Bias \}$

(Objective is Reducing Variance with low bias)

BOOSTING

- Boosting = Base Model + Additively Combining

  $\{\downarrow Variance + \uparrow Bias\}$

  (Objective is Reducing Bias with low Var)

  - High Bias → High training Error
  - If we take a collection of weak learners, each performing only just better than chance, then by putting them together it is possible to make an ensemble learner that can perform arbitrarily well.
  - We need lots of low-quality learners, and a way to put them together usefully, and we can make a learner that will do very well.
  - The principal algorithm of boosting is named
    - Adaptive boosting → Images → Face detection
    - Gradient Boosting → DT (CART)

Step 0:

- D_{train} → M_0 (Consider this as base model) → Weak Learners → $\{\downarrow Variance + \uparrow Bias\}$

  $\{x_i, y_i\}$ --> $y = h_0(x)$ (function to fit the model)
- $y_i - h_0(x_i) = Error_{i0}$
- $\{ x_i, y_i, error_i \}$

Step 1:

- D_{train} → M_1 (Consider this as base model) → Weak Learners → $\{\downarrow Variance + \uparrow Bias\}$

  $\{x_i, error_i\}$ --> $y = h_1(x)$ (function to fit the model) $y_i - h_1(x_i) = Error_1$
- Model 1 training on $error_i$ not y_i
- End of stage 1 → $F_1(x)$
- $F_1(x) = \alpha 0\ h0(x) + \alpha 1\ h1(x)$ [sum of two models]

Step 2:

- D_{train} → M_2 → $h_2(x)$ → $y_i - F_1(x_i)$
- $F_2(x) = \alpha 0\ h0(x) + \alpha 1\ h1(x) + \alpha 2\ h2(x)$

Step K:

- End of Stage k :- $F_k(x) = \sum_{i=0}^{k} \alpha i\ hi(x)$
- Additive Weighted Model -> Residual error => $h_i(x)$ ← { X_i, $error_i$}

  As k increases eventually training error reduces, this will leads to reduce the bias.

  AdaBoosting – Geometric Intuition

  - Weightage given to misclassified point -> Adaboosting
  - At every stage you are adapting to the errors that are made in the prev stage.
  - Decision Tree with depth = 1 : Depth → Decision Stump -> parallel to y or x axis, D2 become dataset for h2.

Given: n examples $(\mathbf{x}_i, y_i)$, where $\mathbf{x} \in \mathcal{X}, y \in \pm 1$.

Initialize: $D_1(i) = \frac{1}{n}$

For $t = 1 \ldots T$

- Train weak classifier on distribution $D(i)$, $h_t(\mathbf{x}) : \mathcal{X} \mapsto \pm 1$
- Choose weight α_t (see how below)
- Update: $D_{t+1}(i) = \frac{D_t(i) \exp\{-\alpha_t y_i h_t(\mathbf{x}_i)\}}{Z_t}$, for all i, where $Z_t = \sum_i D_t(i) \exp\{-\alpha_t y_i h_t(\mathbf{x}_i)\}$

Output classifier: $h(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

Choosing α_t

At iteration t, we obtain a classifier $h_t(\mathbf{x})$, whose weighted error is defined as:

$$\epsilon_t = \sum_i D_t(i) \mathbf{1}(y_i \neq h_t(\mathbf{x_i}))$$

$$\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$$

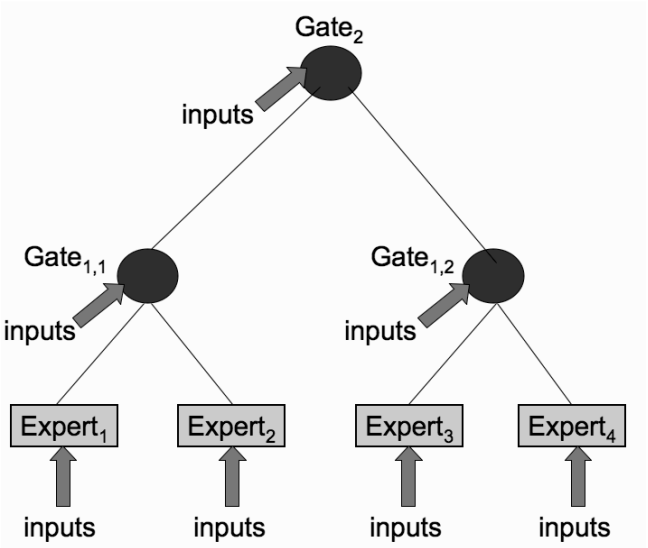
DIFFERENT WAYS TO COMBINE CLASSIFIERS

- Both boosting and bagging take a vote from amongst the classifiers, although they do it in different ways:
- Boosting takes a weighted vote
- Bagging takes the majority vote.
- Assuming that each individual classifier has a success rate of p, the probability of the ensemble getting the correct answer is a binomial distribution of the form: "T" is no. of classifiers.

$$\sum_{k=T/2+1}^{T} \binom{T}{k} p^k (1-p)^{T-k},$$

Mixture of experts

- Inputs are presented to the network, and each individual classifier makes an assessment.
- These outputs from the classifiers are then weighted by the relevant gate, which produces a weight w using the current inputs, and this is propagated further up the hierarchy..

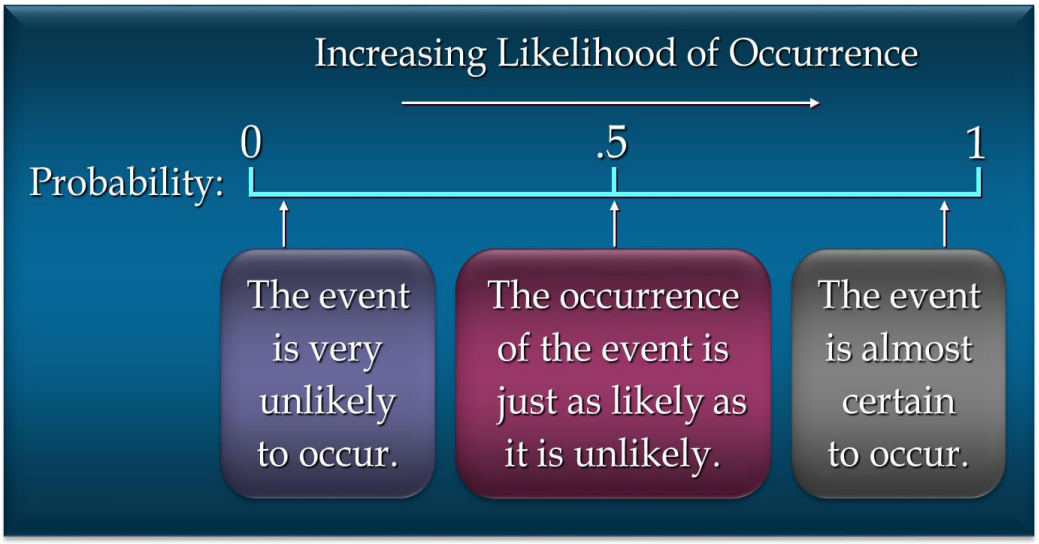


PROBABILITY AND LEARNING – DATA INTO PROBABILITIES – BASIC STATISTICS

- Machine Learning is an interdisciplinary field that uses statistics, probability, algorithms to learn from data and provide insights which can be used to build intelligent applications.
- Probability is a numerical measure of the likelihood that an event will occur.
- Definition Of Theoretical Probability

  $$P(E) = \frac{\text{No. of Favorable outcomes}}{\text{Total no. of outcomes}}$$

  Where E – event
- Probability values are always assigned on a scale from 0 to 1.
- A probability near zero indicates an event is quite unlikely to occur.
- A probability near one indicates an event is almost certain to occur.



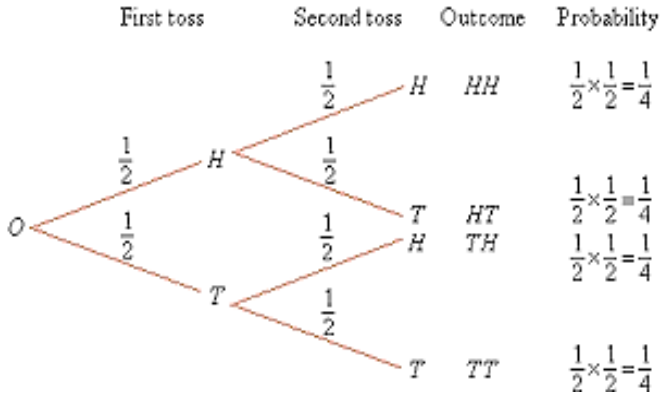
- Experimental Probability is the chance of something happening, based on repeated testing and observing results.
- It is the ratio of the number of times an event has occurred to the number of times it has been tested.
- For example to find the probability of getting a six when rolling a dice
- You need to roll the dice many times

- Divide the number of times who rolled a six with how many times you rolled the dice in total.
- Some common terms related to probability :
- Sample space: The set of outcomes of an experiment is known as sample space.
- Event: One or more outcomes in an experiment.
- Sample point: Each element of the sample space is called a sample point.

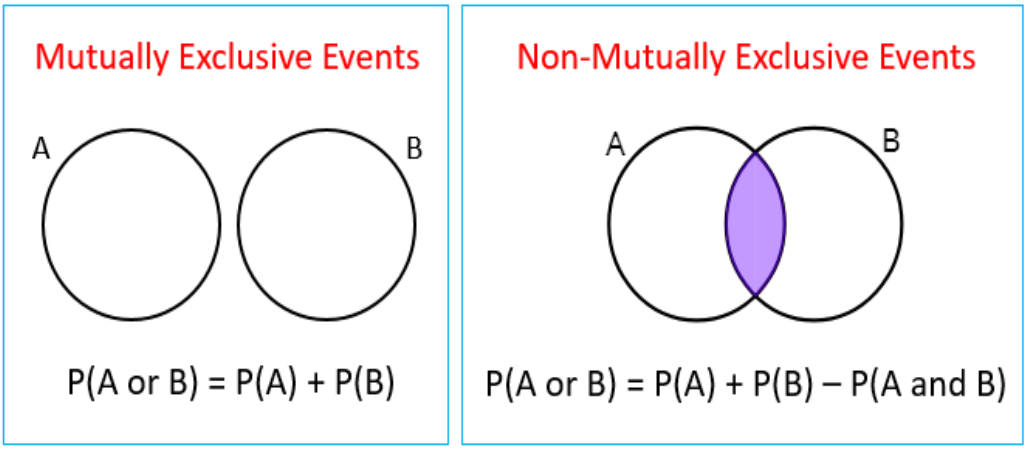
Sample Space

A **sample space** is the set of all possible outcomes in an experiment.

Example:
Two coins are tossed. Represent the sample space for this experiment by making a list, a table, and a tree diagram.
(H – Head, T – Tail)



- Two events are said to be **mutually exclusive** if they cannot happen at the same time.



Mutually Exclusive Example

What is the probability of a dice showing a 2 or 5?

$$P(2)=\frac{1}{6} \qquad P(5)=\frac{1}{6}$$

$$P(2 \text{ or } 5)=P(2)+P(5)$$

$$=\frac{1}{6}+\frac{1}{6}$$

$$=\frac{2}{6}=\frac{1}{3}$$

The probability of a dice showing 2 or 5 is $\frac{1}{3}$

- **Independent Events:**
  - Events are **independent** if the outcome of one event does not affect the outcome of another.
  - For example, if you throw a die and a coin, the number on the die does not affect whether the result you get on the coin.
  - How to calculate the probability of independent events?
    - If A and B are independent events, then the probability of A happening AND the probability of B happening is P(A) × P(B).

Probability of Independent Events

When two events, A and B, are independent the probability of both occurring is:

P(A and B) = P(A) x P(B)

Example:

If a dice is thrown twice, find the probability of getting two 5's.

Solution:

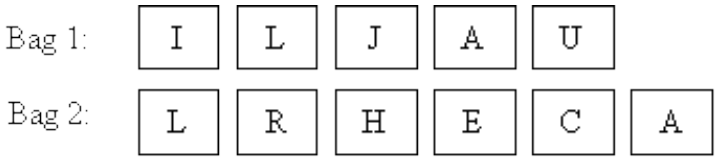
$$P(\text{getting a 5 on the second throw}) = \frac{1}{6}$$

$$P(\text{getting a 5 on the first throw}) = \frac{1}{6}$$

$$P(\text{two 5's}) = \frac{1}{6} \times \frac{1}{6} = \frac{1}{36}$$

Example:

Two sets of cards with a letter on each card as follows are placed into separate bags.



Sara randomly picked one card from each bag. Find the probability that:

a) She picked the letters ‘J’ and ‘R’.

b) Both letters are ‘L’.

c) Both letters are vowels.

Solution:

a) Probability that she picked J and R = $\frac{1}{5}\times\frac{1}{6}=\frac{1}{30}$

b) Probability that both letters are L = $\frac{1}{5}\times\frac{1}{6}=\frac{1}{30}$

c) Probability that both letters are vowels = $\frac{3}{5}\times\frac{2}{6}=\frac{3}{15}=\frac{1}{5}$

Dependent Events

- Events are dependent if the outcome of one event affects the outcome of another.
- If A and B are dependent events, then the probability of A happening AND the probability of B happening, given A, is P(A) × P(B after A).
- P(A and B) = P(A) × P(B after A)
- P(B after A) can also be written as P(B | A)
- then P(A and B) = P(A) × P(B | A)

Independent Events

The outcome of one event does not affect the outcome of the other.

If A and B are independent events then the probability of both occurring is

$$P(A \text{ and } B) = P(A) \times P(B)$$

Dependent Events

The outcome of one event affects the outcome of the other.

If A and B are dependent events then the probability of both occurring is

$$P(A \text{ and } B) = P(A) \times P(B|A)$$

Probability of B given A

- Example: A purse contains.
  - four $5 bills,
  - five $10 bills **4+5+3 =12 bills**
  - Three $20 bills.
  - Two bills are selected without the first selection being replaced.
  - Find P($5, then $5)

Solution:

- There are four $5 bills and there are a total of twelve bills.
- P($5) = 4/12
- The result of the first draw affected the probability of the second draw.
- There are three $5 bills left.
- There are a total of eleven bills left.
- P($5 after $5) = 3/11

P($5 and $5) = P($5) * P($5 after $5) = $\frac{4}{12} * \frac{3}{11} \rightarrow \frac{1}{11}$

- The probability of an event occurring given that another event has already occurred is called a conditional probability.
- Recall that when two events, A and B, are dependent, the probability of both occurring is:
- P(A and B) = P(A) × P(B given A)

  or
- P(A and B) = P(A) × P(B | A)
- If we divide both sides of the equation by P(A) we get the
- Formula for Conditional Probability

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

$$P(B \mid A) = \frac{P(A \text{ and } B)}{P(A)}$$

Example:

- Susan took two tests. The probability of her passing both tests is 0.6. The probability of her passing the first test is 0.8. What is the probability of her passing the second test given that she has passed the first test?

Solution:

$$P(\text{second} \mid \text{first}) = \frac{P(\text{first and second})}{P(\text{first})} = \frac{0.6}{0.8} = 0.75$$

Basic Statistics

Statistics is defined as, collection, presentation, analysis and interpretation of numerical data.

Statistical Description of Data

- ☐ Statistics describes a numeric set of data by its
    - ☐ Center
    - ☐ Variability
    - ☐ Shape
- ☐ Statistics describes a categorical set of data by
    - ☐ Frequency, percentage or proportion of each category

Cases, Variables

- A data set contains information about a sample.
- A Dataset consists of cases.
- Cases are nothing but the objects in the collection.
- Each case has one or more attributes or qualities, called variables which are characteristics of cases.

  Example:
- Suppose you are collecting information about breast cancer patients. Now for each and every cancer patient you want to know the following information

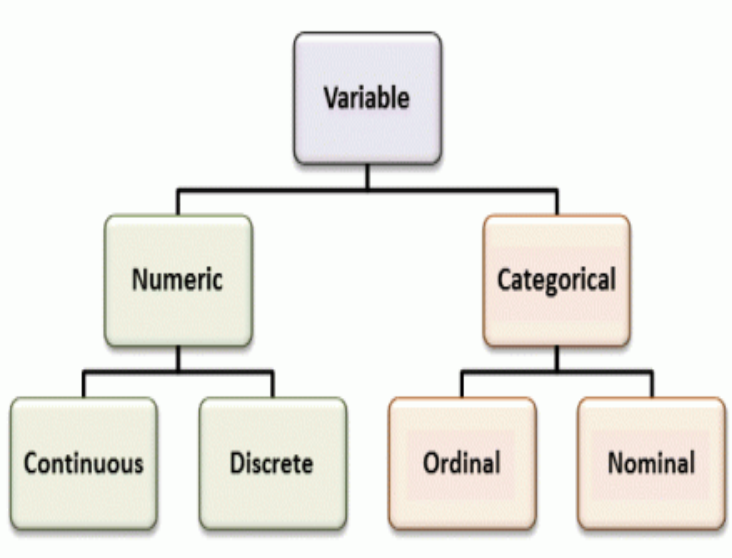
  Categorical Variables:
- Both nominal and ordinal variables can be called categorical variables.

  1. Nominal Variable:
- A nominal variable is made up of various categories which has no order.

  Example:
- Gender of a patient may be Male or Female or State where they live in. Here each category differs from each other but there is no ranking order.

2. Ordinal Variable:

- The second level of measurement is the ordinal level. There is not only a difference between the categories of a variable; there is also an order. An example might be Highest paid, Average Paid and Lowest Paid employee.



Quantitative/ Numerical Variables:

1. Continuous Variable:

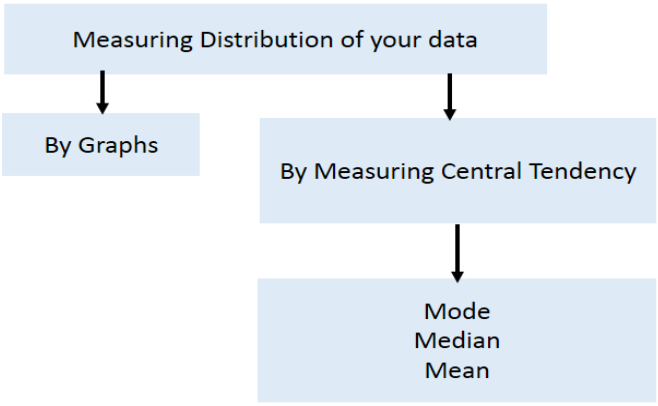
A variable is continuous if the possible values of the variable form an interval. An example is, again, the height of a patient. Someone can be 172 centimeters tall and 174 centimeters tall. But also, for instance, 170.2461. We don't have a set of separate numbers, but an infinite region of values.

2. Discrete Variable:

A variable is discrete if its possible categories form a set of separate numbers. For the above breast cancer data Uniformity of Cell Size: 1 – 10 is an example of discrete variable.

Measuring central tendency of a variable then 3 M's come into picture.

1. Mode
2. Median
3. Mean



MODE, MEDIAN AND MEAN

Find the mean, median, mode, and range for the following list of values:

13, 18, 13, 14, 13, 16, 14, 21, 13

1. The mean is the usual average, so add and then divide:

   (13 + 18 + 13 + 14 + 13 + 16 + 14 + 21 + 13) ÷ 9 = 15

2. The median is the middle value, first rewrite the list in numerical order:

   13, 13, 13, 13, 14, 14, 16, 18, 21

   There are nine numbers in the list, so the middle one will be the (9 + 1) ÷ 2 = 10 ÷ 2 = 5th number:

   13, 13, 13, 13, 14, 14, 16, 18, 21

   So the median is 14.

3. The mode is the number that is repeated more often than any other,

   13 is the mode.

4. The largest value in the list is 21, and the smallest is 13,

   range is 21 – 13 = 8.

- ❖ mean: 15
- ❖ median: 14
- ❖ mode: 13
- ❖ range: 8

GAUSSIAN MIXTURE MODELS

Gaussian mixture models are a probabilistic model for representing normally distributed subpopulations within an overall population. Mixture models in general don't require knowing which subpopulation a data point belongs to, allowing the model to learn the subpopulations automatically. Since subpopulation assignment is not known, this constitutes a form of unsupervised learning.

For example, in modeling human height data, height is typically modeled as a normal distribution for each gender with a mean of approximately 5'10" for males and 5'5" for females. Given only the height data and not the gender assignments for each data point, the distribution of all heights would follow the sum of two scaled (different variance) and shifted (different mean) normal distributions. A model making this assumption is an example of a Gaussian mixture model (GMM), though in general a GMM may have more than two components. Estimating the parameters of the individual normal distribution components is a canonical problem in modeling data with GMMs.

GMMs have been used for feature extraction from speech data, and have also been used extensively in object tracking of multiple objects, where the number of mixture components and their means predict object locations at each frame in a video sequence.

Maximum Likelihood over a GMM

- As usual: Identify a likelihood function

$$\ln p(x|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k N(x_n|\mu_k,\Sigma_k) \right\}$$

- Optimization of means.

$$\ln p(x|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k N(x_n|\mu_k,\Sigma_k) \right\}$$

$$\frac{\partial \ln p(x|\pi,\mu,\Sigma)}{\partial \mu_k} = \sum_{n=1}^{N} \frac{\pi_k N(x_n|\mu_k,\Sigma_k)}{\sum_j \pi_j N(x_n|\mu_j,\Sigma_j} \Sigma_k^{-1}(x_k-\mu_k) = 0$$

$$= \sum_{n=1}^{N} \tau(z_{nk})\Sigma_k^{-1}(x_k-\mu_k) = 0$$

$$\boxed{\mu_k = \frac{\sum_{n=1}^{N} \tau(z_{nk}) x_n}{\sum_{n=1}^{N} \tau(z_{nk})}}$$

- Optimization of covariance

$$\ln p(x|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k N(x_n|\mu_k,\Sigma_k) \right\}$$

$$\boxed{\Sigma_k = \frac{1}{\sum_{n=1}^{N} \tau(z_{nk})} \sum_{n=1}^{N} \tau(z_{nk})(x_k-\mu_k)(x_k-\mu_k)^T}$$

- Optimization of mixing term

$$\ln p(x|\pi,\mu,\Sigma) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right)$$

$$0 = \sum_{n=1}^{N} \frac{\pi_k N(x_n|\mu_k,\Sigma_k)}{\sum_j \pi_j N(x_n|\mu_j,\Sigma_j} + \lambda$$

$$\boxed{\pi_k = \frac{\sum_{n=1}^{N} \tau(z_n k)}{N}}$$

The EM algorithm for Gaussian mixtures

In the case of Gaussian mixture problems, because of the nature of the function, finding a maximum likelihood estimate by taking the derivatives of the log-likelihood function with respect to all the parameters and simultaneously solving the resulting equations is nearly impossible. So we apply the EM algorithm to solve the problem. As already indicated, the EM algorithm is a general procedure for estimating the parameters in a statistical model. This algorithm can be adapted to develop an algorithm for estimating the parameters in a Gaussian mixture model. The adapted EM algorithm has been explained below.

- E-step: Evaluate the Responsibilities

$$\tau(z_{nk}) = \frac{\pi_k N(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n|\mu_j, \Sigma_j)}$$

- M-Step: Re-estimate Parameters

$$\mu_k^{new} = \frac{\sum_{n=1}^{N} \tau(z_{nk}) x_n}{N_k}$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \tau(z_{nk})(x_k - \mu_k^{new})(x_k - \mu_k^{new})^T$$

$$\pi_k^{new} = \frac{N_k}{N}$$

NEAREST NEIGHBOR METHODS

- Learning Algorithm:
  - Store training examples
- Prediction Algorithm:
  - To classify a new example **x** by finding the training example ($\mathbf{x}^i$,y^i) that is *nearest* to **x**
  - Guess the class $y = y^i$

K Nearest Neighbors - Classification

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

Algorithm

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

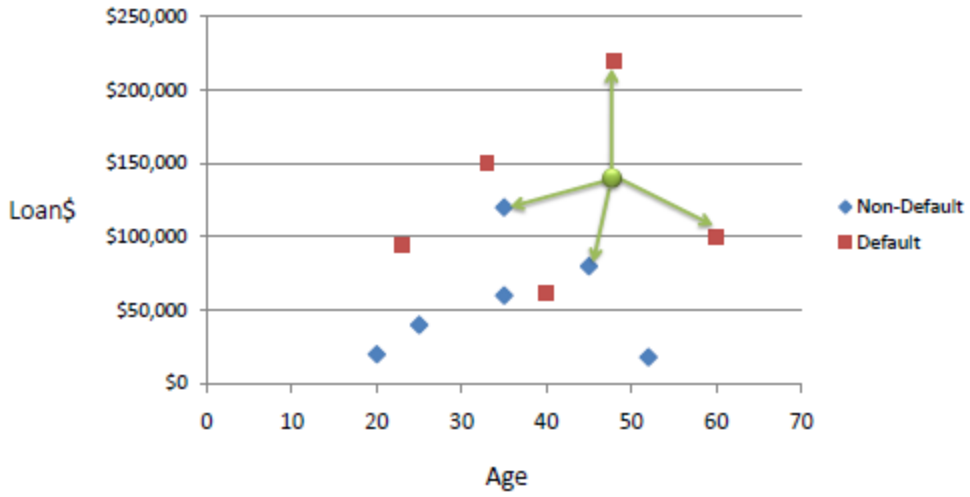
Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee. Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.

Example:

Consider the following data concerning credit default. Age and Loan are two numerical variables (predictors) and Default is the target.



We can now use the training set to classify an unknown case (Age=48 and Loan=$142,000) using Euclidean distance. If K=1 then the nearest neighbor is the last case in the training set with Default=Y.

D = Sqrt[(48-33)^2 + (142000-150000)^2] = 8000.01 >> Default=Y

Age	Loan	Default	Distance	
25	$40,000	N	102000	
35	$60,000	N	82000	
45	$80,000	N	62000	
20	$20,000	N	122000	
35	$120,000	N	22000	2
52	$18,000	N	124000	
23	$95,000	Y	47000	
40	$62,000	Y	80000	
60	$100,000	Y	42000	3
48	$220,000	Y	78000	
33	$150,000	Y	8000	1
48	**$142,000**	**?**		

Euclidean Distance

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

With K=3, there are two Default=Y and one Default=N out of three closest neighbors. The prediction for the unknown case is again Default=Y.

UNSUPERVISED LEARNING-

Unsupervised learning: The data have no target attribute.

We want to explore the data to find some intrinsic structures in them.

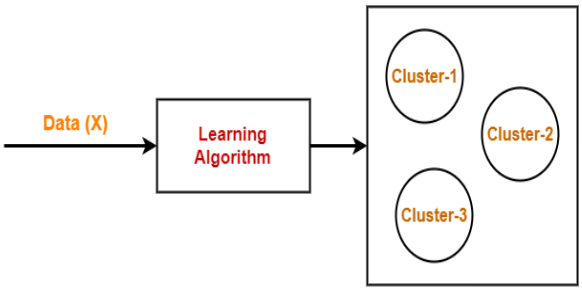
In this type of machine learning algorithm,

- The training data set is an unlabeled data set.
- In other words, the training data set contains only the input value (X) and not the target value (Y).
- Based on the similarity between data, it tries to draw inference from the data such as finding patterns or clusters.

Applications-

Some real-life applications are-

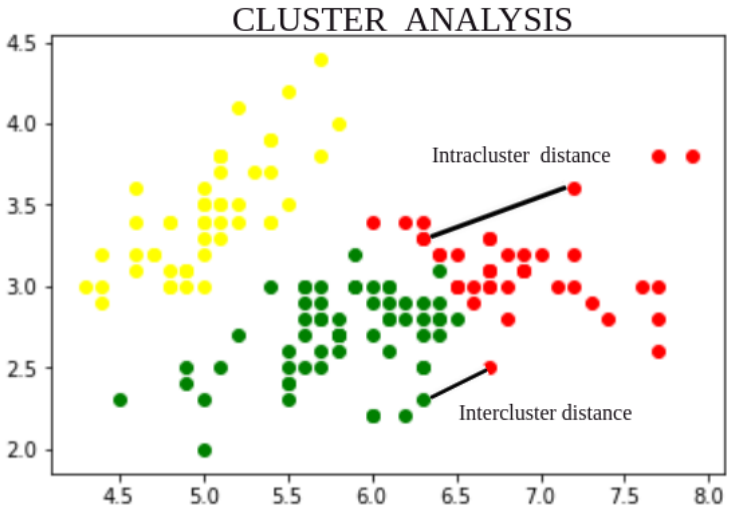
- Document Clustering, Finding fraudulent transactions



Unsupervised Learning

K MEANS ALGORITHMS

- Given a set of n distinct objects, the k-Means clustering algorithm partitions the objects into k number of clusters such that intracluster similarity is high but the intercluster similarity is low.
- In this algorithm, user has to specify k, the number of clusters and consider the objects are defined with numeric attributes and thus using any one of the distance metric to demarcate the clusters.



- First it selects *k* number of objects at random from the set of n objects. These *k* objects are treated as the centroids or center of gravities of *k* clusters.
- For each of the remaining objects, it is assigned to one of the closest centroid. Thus, it forms a collection of objects assigned to each centroid and is called a cluster.
- Next, the centroid of each cluster is then updated (by calculating the mean values of attributes of each object).
- The assignment and update procedure is until it reaches some stopping criteria (such as, number of iteration, centroids remain unchanged or no assignment, etc.)

Algorithm : k-Means clustering

Input: D is a dataset containing *n* objects, *k* is the number of cluster

Output: A set of *k* clusters

Steps:

1. Randomly choose *k* objects from D as the initial cluster centroids.
2. **For** each of the objects in D **do**
    - Compute distance between the current objects and *k* cluster centroids
    - Assign the current object to that cluster to which it is closest.
    - Compute the "cluster centers" of each cluster. These become the new cluster centroids.
3. Repeat step 2-3 until the convergence criterion is satisfied
4. Stop

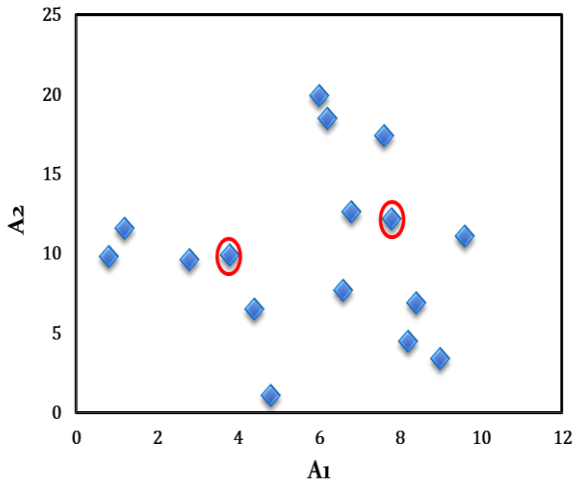
Note:

1) Objects are defined in terms of set of attributes. $A = \{A_1, A_2, \ldots\ldots, A_m\}$ where each A_i is continuous data type.
2) Distance computation: Any distance such as L_1, L_2, L_3 or cosine similarity.

3) Minimum distance is the measure of closeness between an object and centroid.
4) Mean Calculation: It is the mean value of each attribute values of all objects.
5) Convergence criteria: Any one of the following are termination condition of the algorithm.
    - Number of maximum iteration permissible.
    - No change of centroid values in any cluster.
    - Zero (or no significant) movement(s) of object from one cluster to another.
    - Cluster quality reaches to a certain level of acceptance.

Table: objects with two attributes A_1 andA_2.

Plotting data



xi	A_1	A_2
X1	6.8	12.6
X2	0.8	9.8
X3	1.2	11.6
X4	2.8	9.6
X5	3.8	9.9
X6	4.4	6.5
X7	4.8	1.1
X8	6.0	19.9
X9	6.2	18.5
X10	7.6	17.4
X11	7.8	12.2
X12	6.6	7.7
X13	8.2	4.5
X14	8.4	6.9
X15	9.0	3.4
X16	9.6	11.1

- Suppose, k=3. Three objects are chosen at random shown as circled (see Fig 16.1). These three centroids are shown below.

Initial Centroids chosen randomly

Centroid	Objects	
	A1	A2
c_1	3.8	9.9
c_2	7.8	12.2
c_3	6.2	18.5

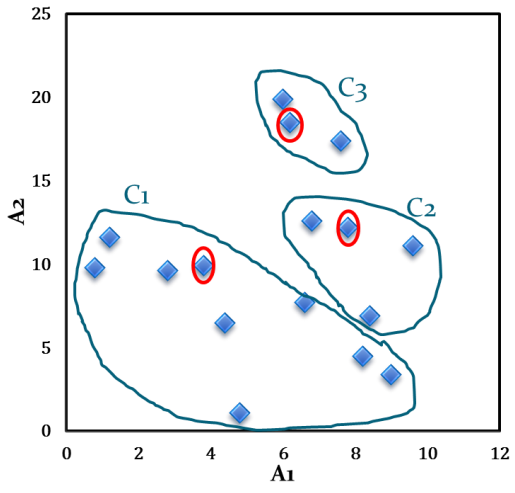
- Let us consider the Euclidean distance measure (L_2 Norm) as the distance measurement in our illustration.

- Let d_1, d_2 and d_3 denote the distance from an object to c_1, c_2 and c_3 respectively. The distance calculations are shown in Table 16.2.
- Assignment of each object to the respective centroid is shown in the right-most column and the clustering so obtained is shown in Fig 16.2.

Table 16.2: Distance calculation

	A_1	A_2	d_1	d_2	d_3	cluster
X1	6.8	12.6	4.0	1.1	5.9	2
X2	0.8	9.8	3.0	7.4	10.2	1
X3	1.2	11.6	3.1	6.6	8.5	1
X4	2.8	9.6	1.0	5.6	9.5	1
X5	3.8	9.9	0.0	4.6	8.9	1
X6	4.4	6.5	3.5	6.6	12.1	1
X7	4.8	1.1	8.9	11.5	17.5	1
X8	6.0	19.9	10.2	7.9	1.4	3
X9	6.2	18.5	8.9	6.5	0.0	3
X10	7.6	17.4	8.4	5.2	1.8	3
X11	7.8	12.2	4.6	0.0	6.5	2
X12	6.6	7.7	3.6	4.7	10.8	1
X13	8.2	4.5	7.0	7.7	14.1	1
X14	8.4	6.9	5.5	5.3	11.8	2
X15	9.0	3.4	8.3	8.9	15.4	1
X16	9.6	11.1	5.9	2.1	8.1	2

Fig 16.2: Initial cluster with respect to Table 16.2



Euclidean distance

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

x1, y1 x2 y2

(6.8, 12.6) & (3.8, 9.9) C1

Sqrt(9 + 7.29)➔ sqrt(16.29)

	A_1	A_2	cluster
X2	0.8	9.8	1
X3	1.2	11.6	1
X4	2.8	9.6	1
X5	3.8	9.9	1
X6	4.4	6.5	1
X7	4.8	1.1	1
X12	6.6	7.7	1
X13	8.2	4.5	1
X15	9	3.4	1

	A_1	A_2	cluster
X1	6.8	12.6	2
X11	7.8	12.2	2
X14	8.4	6.9	2
X16	9.6	11.1	2

	A_1	A_2	cluster
X8	6	19.9	3
X9	6.2	18.5	3
X10	7.6	17.4	3

Cluster 2 updated Center
Mean (A1) = 32.6/4 ➔ 8.2
Mean (A2) = 42.8/4 ➔ 10.7

Cluster 3 updated Center
Mean (A1) = 19.8/3 ➔ 6.6
Mean (A2) = 55.8/3 ➔ 18.6

Cluster 1 –updated Center
Mean (A1) = 41.6/9 ➔ 4.6
Mean (A2) = 64.1/9 ➔ 7.1

The calculation new centroids of the three cluster using the mean of attribute values of A_1 and A_2 is shown in the Table below. The cluster with new centroids are shown in Fig 16.3.

Calculation of new centroids

New Centroid	Objects	
	A1	A2
c_1	4.6	7.1
c_2	8.2	10.7
c_3	6.6	18.6

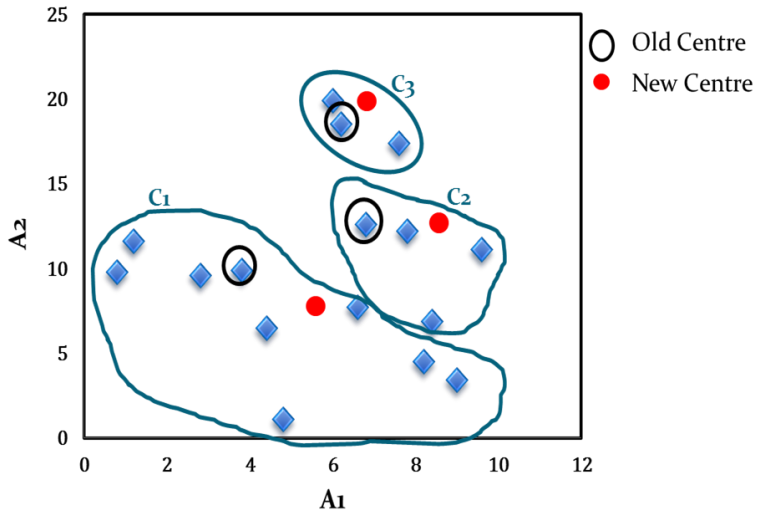


Fig 16.3: Initial cluster with new centroids

We next reassign the 16 objects to three clusters by determining which centroid is closest to each one. This gives the revised set of clusters shown in Fig 16.4.

Note that point p moves from cluster C_2 to cluster C_1.

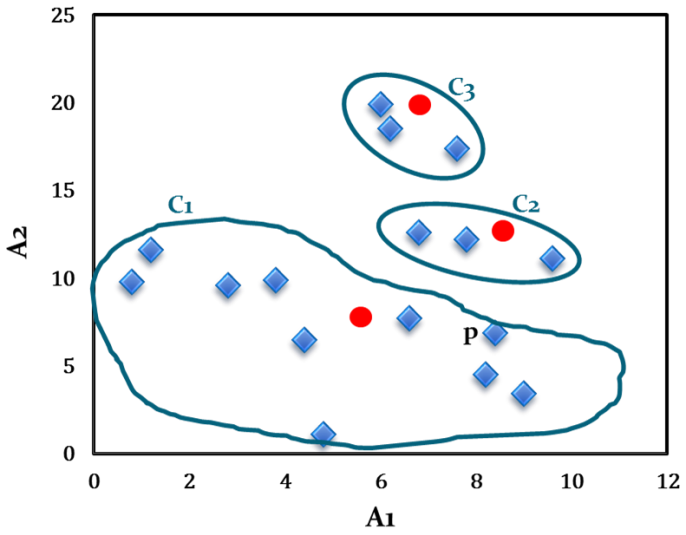


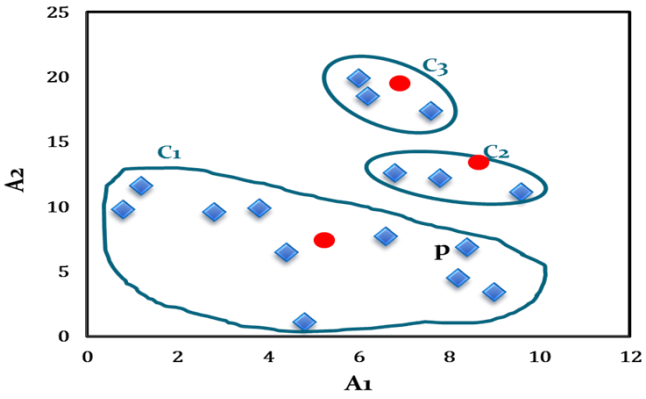
Fig 16.4: Cluster after first iteration

- The newly obtained centroids after second iteration are given in the table below. Note that the centroid c_3 remains unchanged, where c_2 and c_1 changed a little.
- With respect to newly obtained cluster centres, 16 points are reassigned again. These are the same clusters as before. Hence, their centroids also remain unchanged.
- Considering this as the termination criteria, the k-means algorithm stops here. Hence, the final cluster in Fig 16.5 is same as Fig 16.4.

Cluster centres after second iteration

Centroid	Revised Centroids	
	A1	A2
c_1	5.0	7.1
c_2	8.1	12.0
c_3	6.6	18.6

Fig 16.5: Cluster after Second iteration



Let us analyse the k-Means algorithm and discuss the pros and cons of the algorithm.

We shall refer to the following notations in our discussion.

- Notations:
  - x : an object under clustering
  - n : number of objects under clustering
  - $\boldsymbol{C}_i$: the *i-th* cluster
  - c_i : the centroid of cluster $\boldsymbol{C}_i$
  - n_i : number of objects in the cluster $\boldsymbol{C}_i$
  - c : denotes the centroid of all objects
  - k : number of clusters

The k-Means Algorithm

- **Initialisation**
  - choose a value for k
  - choose k random positions in the input space
  - assign the cluster centres μ_j to those positions
- **Learning**
  - repeat
    - for each datapoint $\mathbf{x}_i$:
      - compute the distance to each cluster centre
      - assign the datapoint to the nearest cluster centre with distance

$$d_i = \min_j d(\mathbf{x}_i, \mu_j). \tag{14.1}$$

    - for each cluster centre:
      - move the position of the centre to the mean of the points in that cluster (N_j is the number of points in cluster j):

$$\mu_j = \frac{1}{N_j} \sum_{i=1}^{N_j} \mathbf{x}_i \tag{14.2}$$

  - until the cluster centres stop moving
- **Usage**
  - for each test point:
    - compute the distance to each cluster centre
    - assign the datapoint to the nearest cluster centre with distance

$$d_i = \min_j d(\mathbf{x}_i, \mu_j). \tag{14.3}$$

VECTOR QUANTIZATION

- Data compression is a process of encoding data so that it takes lesser storage space and lesser transmission time than the data which is not compressed.
- Data compression is the art or science of representing information in a compact form.
- Compression is possible because most real world data is very redundant.

COMPRESSION TECHNIQUES

1) Lossless Compression:

– Lossless compression techniques involve no loss of information.

If the data have been lossless compressed, the original data can be recovered exactly from the compressed data.

2) Lossy Compression:

– Lossy compression techniques involve some loss of information, and data that have been compressed using lossy techniques generally cannot be recovered exactly.

TYPES OF LOSLESS DATA COMPRESSION

- Dictionary coders
  – Zip (file formats)
- Entropy coding
  – Huffman coding (simple entropy coding)
- Run-length coding

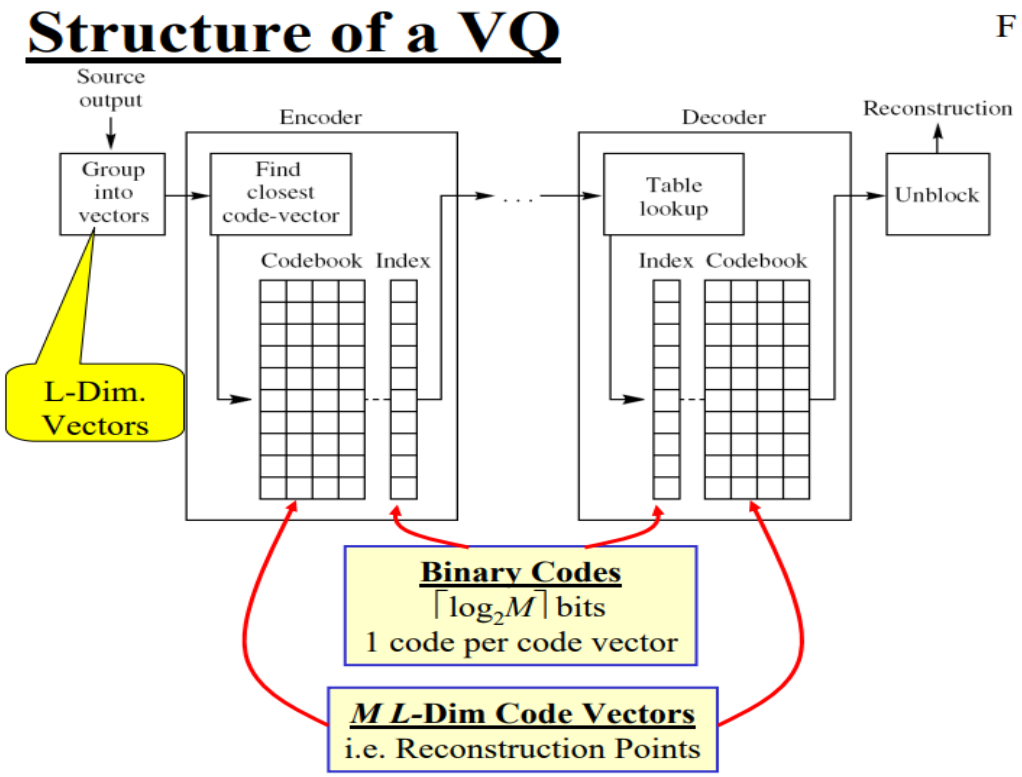
TYPES OF LOSSY DATA COMPRESSION

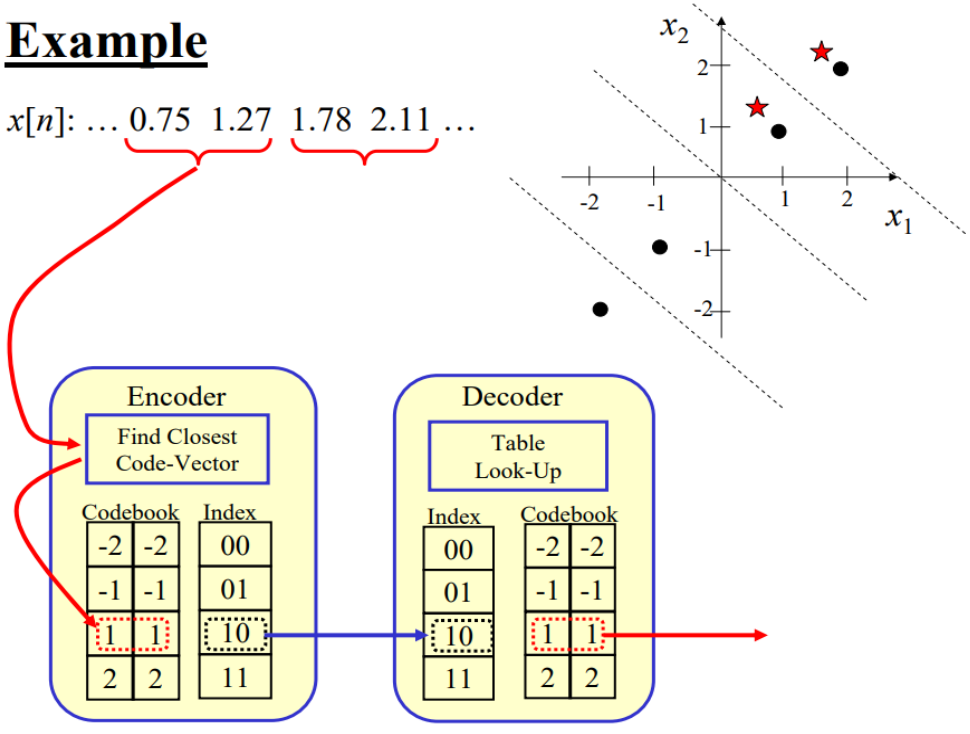
- Scalar Quantization:

– The most common type of quantization is scalar quantization. Scalar quantization, typically denoted as $y = Q(x)$ is the process of using quantization function $Q( )$ to map a scalar input value x to scalar output value y.

- Vector Quantization:

– A vector quantizer maps k-dimensional vectors in the vector space R^k into a finite set of vectors Y = {yi : i = 1, 2, ..., N}. Each vector yi is called a code vector or a codeword. Set of all the codewords is called a codebook.





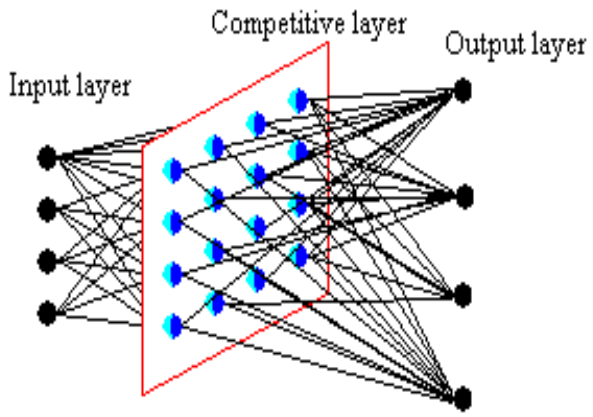
APPLICATION OF VECTOR QUANTIZATION

Vector quantization technique is efficiently used in various areas of biometric modalities like

- finger print pattern recognition ,
- face recognition by generating codebooks of desired size.

SELF-ORGANIZING FEATURE MAP

- Unsupervised training:- in which the networks learn to form their own classifications of the training data without external help.
- To do this we have to assume that class membership is broadly defined by the input patterns sharing common features, and that the network will be able to identify those features across the range of input patterns.
- One particularly interesting class of unsupervised system is based on competitive learning, in which the output neurons compete amongst themselves to be activated, with the result that only one is activated at any one time.
- This activated neuron is called a winner-takes-all neuron or simply the winning neuron. Such competition can be induced/implemented by having lateral inhibition connections (negative feedback paths) between the neurons.
- The result is that the neurons are forced to organize themselves. For obvious reasons, such a network is called a Self Organizing Map (SOM).
- The input is connected with each neuron of a lattice.
- Lattice Topology: It determines a neighborhood structure of the neurons.



- We have to find values for the weight vectors of the links from the input layer to the nodes of the lattice, in such a way that adjacent neurons will have similar weight vectors.
- For an input, the output of the neural network will be the neuron whose weight vector is most similar (with respect to Euclidean distance) to that input.
- In this way, each (weight vector of a) neuron is the center of a cluster containing all the input examples which are mapped to that neuron.

An informal description:

- Given: an input pattern x
- Find: the neuron i which has closest weight vector by competition ($w_i^T x$ will be the highest).
- For each neuron j in the neighboruhood $N(i)$ of the winning neuron i:
  - update the weight vector of j.
- The SOM algorithm:
  - Starts with large neighborhood size and gradually reduces it.
  - Gradually reduces the learning rate η.
  - Upon repeated presentations of the training examples, the weight vectors tend to follow the distribution of the examples.
  - This results in a topological ordering of the neurons, where neurons adjacent to each other tend to have similar weights.

There are basically three essential processes:

- competition
- cooperation
- weight adaption

- Competition:
  - Competitive process: Find the best match of input vector *x* with weight vectors:
  - The input space of patterns is mapped onto a discrete output space of neurons by a process of competition among the neurons of the network.

$$i(x) = \arg\min_j \| x - w_j \| \qquad j = 1, 2, \ldots, \ell$$

winning neuron

total number of neurons

- Convergence phase:
  - Fine tune feature map.
  - Must be at least 500 times the number of neurons in the network $\Rightarrow$ thousands or tens of thousands of iterations.
- Choice of parameter values:
  - $\eta(n)$ Maintained on the order of 0.01.
  - $h_{ji(x)}(n)$ contains only the nearest neighborurs of the winning neuron. It eventually reduces to one or zero neighboruring neurons.
- Initialization: choose random small values for weight vectors such that $w_j(0)$ is different for all neurons j.
- Sampling: drawn a sample example x from the input space.
- Similarity matching: find the best matching winning neuron $i(x)$ at step n:

$$i(x) = \arg\min_j \| x(n) - w_j \| \qquad j \in [1, 2, \ldots, \ell]$$

Updating: adjust synaptic weight vectors

$$w_j(n+1) = w_j(n) + \eta(n)\, h_{ij(x)}(n)\left(\mathrm{x} - w_j(n)\right)$$

- Continuation: go to sampling step until no noticeable changes in the feature map are observed.

1. What is the idea of decision tree?

The idea of a decision tree is that we break classification down into set of choices about each feature in turn, starting at the root (base) of the tree and processing down to the leaves, where we receive the classification decision.

2. List the advantages and disadvantages of decision tree.

- Are simple to understand and interpret. People are able to understand decision tree models after a brief explanation.
- Have value even with little hard data. Important insights can be generated based on experts describing a situation (its alternatives, probabilities, and costs) and their preferences for outcomes.
- Allow the addition of new possible scenarios.
- Help determine worst, best and expected values for different scenarios.
- Use a white box model. If a given result is provided by a model.
- Can be combined with other decision techniques.

Disadvantages of decision trees:

- For data including categorical variables with different number of levels, information gain in decision trees is biased in favor of those attributes with more levels.
- Calculations can get very complex, particularly if many values are uncertain and/or if many outcomes are linked.

3. Define (information) Entropy

Information entropy, which describes the amount of impurity in a set of features. The entropy H of a set of probabilities pi is $H(p) = -\Sigma\, pi \log_2 pi$ (where the logarithm is base 2)

4. Define information gain.

Information gain is defined as the entropy of the whole set minus the entropy when a particular feature is chosen. This is defined by

$Gain(S,F) = Entropy(S) - \sum_{f \in Values(F)} |Sf| / |S| \bullet Entropy(Sf)$

Where S is the set of examples, F is a possible feature out of the set of all possible ones, and |Sf| is a count of the number of members of S that have value f for feature F.

5. List some decision tree algorithms.

There are many specific decision-tree algorithms. Notable ones include:

- ID3 (Iterative Dichotomiser 3)
- C4.5 (successor of ID3)
- CART (Classification And Regression Tree)
- CHAID (CHi-squared Automatic Interaction Detector).

  Performs multi-level splits when computing classification trees.

6. Write note on ID3 algorithm.

- Calculate the entropy of every attribute using the data set
- Split the set S into subsets using the attribute for which the resulting entropy (after splitting) is minimum (or, equivalently, information gain is maximum)
- Make a decision tree node containing that attribute
- Recurse on subsets using remaining attributes.

7. What is CART?

Decision trees used in data mining are of two main types:

- Classification tree analysis is when the predicted outcome is the class to which the data belongs.
- Regression tree analysis is when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital).

The term Classification And Regression Tree (CART) analysis is an umbrella term used to refer

to both of the above procedures, first introduced by Breiman et al. Trees used for regression

and trees used for classification have some similarities - but also some differences, such as
the procedure used to determine where to split.

8. What is Gini impurity?

Used by the CART (classification and regression tree) algorithm, Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. Gini impurity can be computed by summing the probability pi of an item with label I being chosen times the probability 1-pi of a mistake in categorizing that item. It reaches its minimum (zero) when all cases in the node fall into a single target category.

9. What is the goal of ensemble methods?

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

10. What are the two families of ensemble methods?

- In averaging methods, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.

  Examples: Bagging methods, Forests of randomized trees

- By contrast, in boosting methods, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble.

  Examples: AdaBoost, Gradient Tree Boosting

11. What is Boosting?

- Boosting is a general ensemble method that attempts to create a strong classifier from a number of weak classifiers.
- This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added.

12. What is stumping?

There is a very extreme form of boosting that is applied to trees. It goes by the descriptive name of stumping. Stumping consists of simply taking the root of the tree and using as the decision maker.

13. What is bagging?

Bootstrap aggregating, also called bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid over fitting. Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach.

14. What is expectation–maximization (EM) algorithm?

In statistics, an expectation–maximization (EM) algorithm is an iterative method to find maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.

15. What is kernel smoother?

A kernel smoother is a statistical technique for estimating a real valued function by using its noisy observations, when no parametric model for this function is known. The estimated function is smooth, and the level of smoothness is set by a single parameter.

This technique is most appropriate for low-dimensional ($p < 3$) data visualization purposes. Actually, the kernel smoother represents the set of irregular data points as a smooth line or surface.

16. What is Gaussian mixture model?

A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.

17. What is vector quantization?

Vector quantization (VQ) is a classical quantization technique from signal processing that allows the modeling of probability density functions by the distribution of prototype vectors. It was originally used for data compression. It works by dividing a large set of points (vectors) into groups having approximately the same number of points closest to them. Each group is represented by its centroid point, as in k-means and some other clustering algorithms.

18. What is k-NN algorithm?

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression.[1] In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

19. What is SOFM?

A self-organizing map (SOM) or self-organizing feature map (SOFM) is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map, and is therefore a method to do dimensionality reduction. Self-organizing maps differ from other artificial neural networks as they apply competitive learning as opposed to error-correction learning (such as back propagation with gradient descent), and in the sense that they use a neighborhood function to preserve the topological properties of the input space.

PART - B

1. Explain ID3 algorithm with example.
2. Explain classification and regression trees with example.
3. Make a decision tree that computes the logical AND function. How does it compare to the perceptron solution?
4. Explain AdaBoost algorithm with example.
5. Explain the different ways to combine classifiers.
6. Explain the following
    a) Turning data into probabilities
    b) Naïve Bayes classifier
7. Explain the important statistical concepts in detail.
8. Explain the Gaussian Mixture models in detail.
9. Explain nearest neighbor methods in detail.
10. Explain the k-means algorithm in detail
11. Explain vector quantization in detail.
12. Explain SOFM in detail.