



Brain Computer Interaction

Feature Extraction

Course Instructors

Dr. Sreeja S R

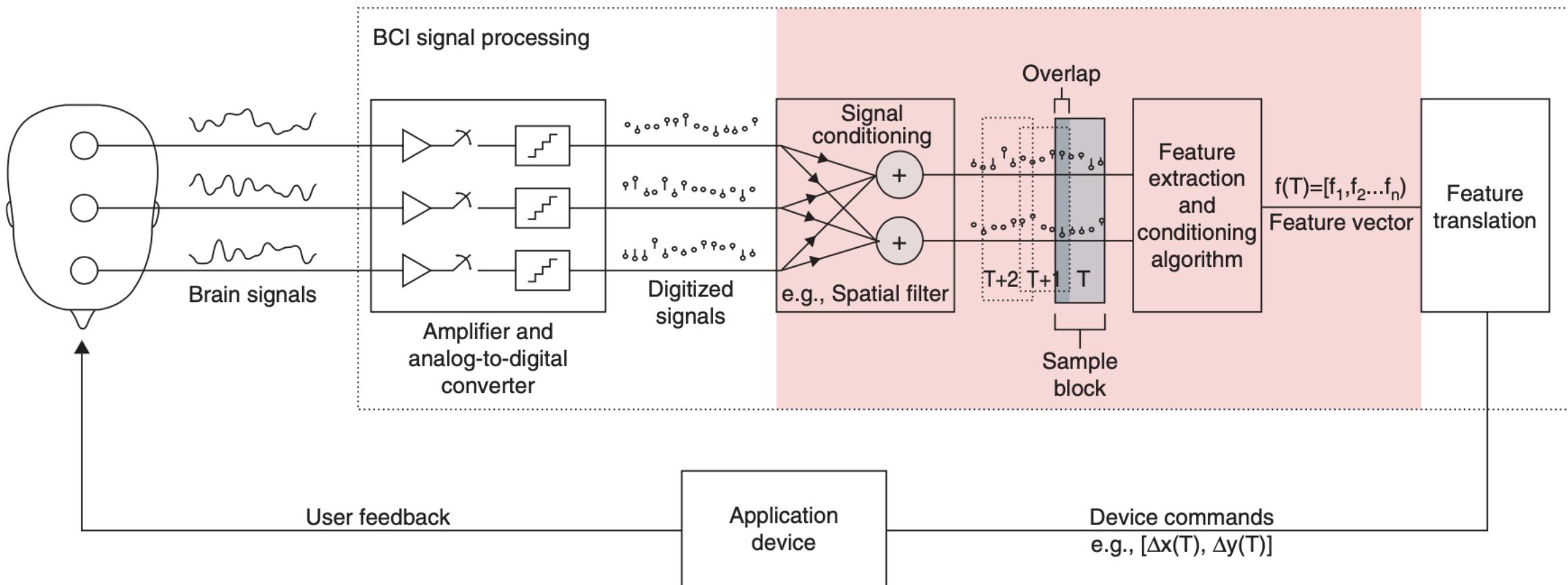
Assistant Professor

Indian Institute of Information Technology
IIIT Sri City

Features

- The purpose of a BCI is to **detect and quantify characteristics of brain signals** that indicate what the user wants the BCI to do, to translate these measurements in real time into the desired device commands, and to provide concurrent feedback to the user.
- The brain-signal characteristics used for this purpose are called ***signal features, or simply features.***
- ***Feature extraction*** is the process of distinguishing the pertinent signal characteristics from extraneous content and representing them in a compact and/or meaningful form, amenable to interpretation by a human or computer.

Overall structure of a BCI



Feature vector

- A **fundamental signal feature** is simply a direct measurement of the signal. They usually provide limited relevant information about typically complex brain signals.
- Thus, it is more common for BCIs to use features that are **linear or nonlinear combinations, ratios, statistical measures, or other transformations of multiple fundamental features** detected at multiple electrodes and/or multiple time points.
- Such complex features, if selected appropriately, can reflect the user's desires more accurately than the fundamental features themselves.
- Most features used in BCI applications are based on **spatial, temporal, and/or spectral analyses** of brain signals or the relationships among them.
- Furthermore, in order to determine the user's wishes as accurately as possible, most BCIs extract a number of features simultaneously. This set of features is referred to as a ***feature vector***.

Feature vector

To be effective for BCI applications, a feature should have the following attributes:

- its spatial, temporal, spectral characteristics, and dynamics can be precisely characterized for an individual user or population of users
- it can be modulated by the user and used in combination with other features to reliably convey the user's intent
- its correlation with the user's intent is stable over time and/or can be tracked in a consistent and reliable manner

SECOND STEP: EXTRACTING THE FEATURES

- ***BLOCK PROCESSING***

- For most BCI applications, it is highly desirable for the processing to occur in real time. Prior to feature extraction, the incoming signal samples are commonly segmented into consecutive, possibly overlapping, sample blocks.
- A feature vector is created from the signal samples within each individual sample block. The feature vectors from the successive sample blocks are then fed to the translation algorithm, which produces a device command or user feedback corresponding to each sample block or corresponding to sets of consecutive sample blocks.
- For efficient online implementation, the length and overlap of these sample blocks should fit the relevant temporal dynamics of the signal, the feature-extraction method, the nature of the application, and the concurrent user feedback, as well as the available processing power.
 - E.g., BCI cursor control
 - P300 response

SECOND STEP: EXTRACTING THE FEATURES

- ***TIME (TEMPORAL) FEATURES***

1. Peak-Picking and Integration

- Peak-picking simply determines the minimum or maximum value of the signal samples in a specific time block (usually defined relative to a specific preceding stimulus) and uses that value (and possibly its time of occurrence) as the feature(s) for that time block.
- The signal can be averaged or integrated over all or part of the time block to yield the feature(s) for the block. Some form of averaging or integration is typically preferable to simple peak-picking, especially when the responses to the stimulus are known to vary in latency and/or when unrelated higher-frequency activity is superimposed on the relevant feature

SECOND STEP: EXTRACTING THE FEATURES

- ***TIME (TEMPORAL) FEATURES***
- 2. Correlation and Template-Matching
 - The similarity of a response to a predefined template might also be used as a feature.
 - Comparing signals in time-domain
 - $X = [1, 2, 3, 2]$
 - $Y = [2, 3, 2, 0]$
 - Find the cross-correlation

SECOND STEP: EXTRACTING THE FEATURES

Statistical features

Sl. No	Features	Short description
1	MEAN	Mean value
2	STD	Standard deviation
3	MAX VALUE	Maximum positive amplitudes
4	MIN VALUE	Maximum negative amplitudes
5	SKEWNESS	a measure of asymmetry of the distribution
6	KURTOSIS	a measure of flatness of the distribution
7	MEDIAN	the middle value of a set of ordered data

SECOND STEP: EXTRACTING THE FEATURES

Interval or period analysis features.

Sl. No	Features	Short description
8	LINE LENGTH	Line length
9	MEAN VV AMPL	Mean of vertex-to-vertex amplitudes
10	VAR VV AMPL	Variance of vertex-to-vertex amplitudes
11	MEAN VV TIME	Mean of vertex-to-vertex times
12	VAR VV TIME	Variance of vertex-to-vertex times
13	MEAN VV SLOPE	Mean of vertex-to-vertex slope
14	VAR VV SLOPE	Variance of vertex-to-vertex slope
15	ZERO CROSSING	Number of zero crossings in a signal
16	MIN MAX NUMBER	Number of local minima and maxima
17	COEFF OF VARIATION	a statistical measure of the deviation of a variable from its mean, standard deviation divided by mean
18	AMPL RANGE	The difference between the maximum positive and maximum negative Amplitude values

SECOND STEP: EXTRACTING THE FEATURES

Features derived from the first and second derivative.

Sl. No	Features	Short description
19	1 st DIFF MEAN	Mean value of the first derivative of the signal
20	1 st DIFF MAX	Maximum value of the first derivative of the signal
21	2 nd DIFF MEAN	Mean value of the second derivative of the signal
22	2 nd DIFF MAX	Maximum value of the second derivative of the signal

SECOND STEP: EXTRACTING THE FEATURES

The Hjorth parameters

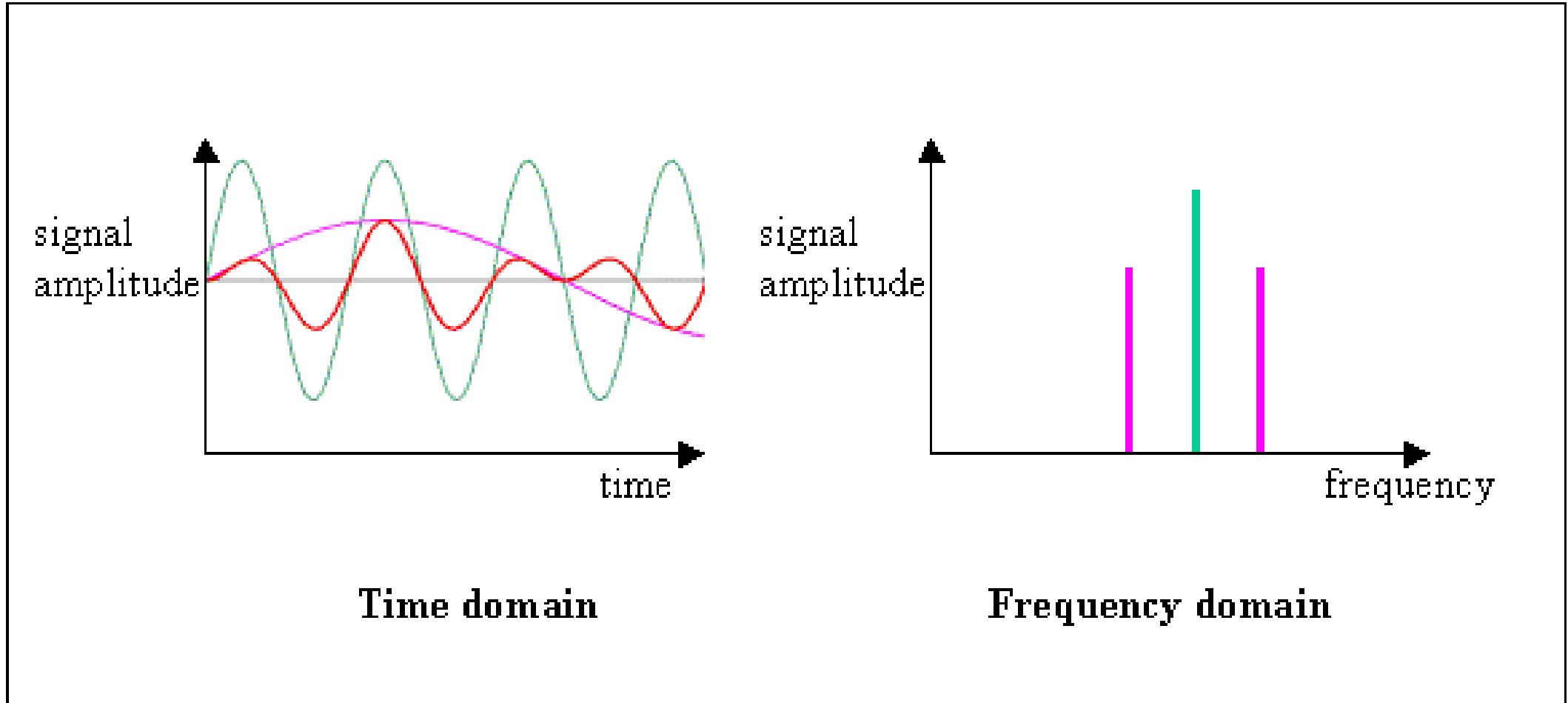
Sl. No	Features	Short description
23	HJORTH 1	Ability
24	HJORTH 2	Mobility $(\sigma x' / \sigma x)$
25	HJORTH 3	Complexity $\frac{(\sigma x' / \sigma x'')}{(\sigma x' / \sigma x)}$

NOTE:

$\sigma x'$ is the standard deviation of the first derivative of the signal
 $\sigma x''$ is the standard deviation of the second derivative of the signal

SECOND STEP: EXTRACTING THE FEATURES

- ***FREQUENCY (SPECTRAL) FEATURES***
- Much brain activity manifests itself as continuous amplitude- and frequency-modulated oscillations. Therefore, it is often advantageous to accurately track these changes in the frequency domain. Although the Fourier transform is the most common method for converting from the time domain to the frequency domain, there are several alternatives that have characteristics that are particularly desirable given specific constraints or specific objectives. These include:
 - band power
 - fast Fourier transform (FFT)
 - Magnitude squared coherence (MSC)
 - Power spectral densities



SECOND STEP: EXTRACTING THE FEATURES

FFT-based features calculated from the EEG spectra.

Sl. No	Features	Short description
26	FFT DELTA	0.1 - 3 Hz
27	FFT THETA	3 - 7 Hz
28	FFT ALPHA	7 - 12 Hz
29	FFT BETA	12 - 30 Hz
30	FFT GAMMA	30 - 40 Hz
31	FFT WHOLE	0.1 - 40 Hz

SECOND STEP: EXTRACTING THE FEATURES

FFT-based Spectral Features.

Sl. No	Features	Short description
32	FFT DT RATIO	<i>DELTA / THETA</i>
33	FFT DA RATIO	<i>DELTA / ALPHA</i>
34	FFT TA RATIO	<i>THETA / ALPHA</i>
35	FFT DTA RATIO	<i>(DELTA + THETA) / ALPHA</i>
36	FFT SEF	Spectral edge frequency (95 % of the total spectral power resides)
37	FFT SP-ROLL OFF	The frequency below which 85 % of the total spectral power resides

Spectral edge frequency 95 (SEF 95 : the frequency below which 95 % of the EEG power is located) was calculated. Spectral bands of 0–4 Hz (delta) 4–8 Hz (theta), 8–13 Hz (alpha) and 13–30 Hz (beta) were analysed and the power of the spectral bands were calculated and expressed as percentage of total power.

SECOND STEP: EXTRACTING THE FEATURES

Wavelet based Features.

Sl. No	Features	Short description
38	MIN WAV VALUE	Minimum value
39	MAX WAV VALUE	Maximum value
40	MEAN WAV VALUE	Mean value
41	MEDIAN WAV VALUE	Median value
42	STD WAV VALUE	Standard deviation
43	SKEWNESS WAV VALUE	Skewness
44	KURTOSIS WAV VALUE	Kurtosis
45	WAV BAND	Relative energy

SECOND STEP: EXTRACTING THE FEATURES

Wavelet based Features.

Sl. No	Features	Short description
46	ENTROPY SPECTRAL WAV	The spectral entropy
47	1 st DIFF WAV MEAN	Mean value of the 1 st derivative
48	1 st DIFF WAV MAX	Maximum value of the 1 st derivative
49	2 nd DIFF WAV MEAN	Mean value of the 2 nd derivative
50	2 nd DIFF WAV MAX	Maximum value of the 2 nd derivative
51	ENERGY PERCENT WAV	Percentage of the total energy of a detail/approximation
52	WAV ZERO CROSSING	Zero crossing
53	WAV COEFF OF VARIATION	Coefficient of variation
54	WAV TOTAL ENERGY	Total Energy

SECOND STEP: EXTRACTING THE FEATURES

Other Features.

Sl. No	Features	Short description
55	ENTROPY SPECTRAL	The spectral entropy
56	ENTROPY SHANNON	The Shannon entropy
57	MAX ABS XCORR EEG-EEG	Maximum positive amplitude of auto-correlation or cross-Correlation function
58	MEAN ABS XCORR EEG-EEG	Mean value of auto-correlation or cross-correlation function

THIRD STEP: FEATURE CONDITIONING / Selection

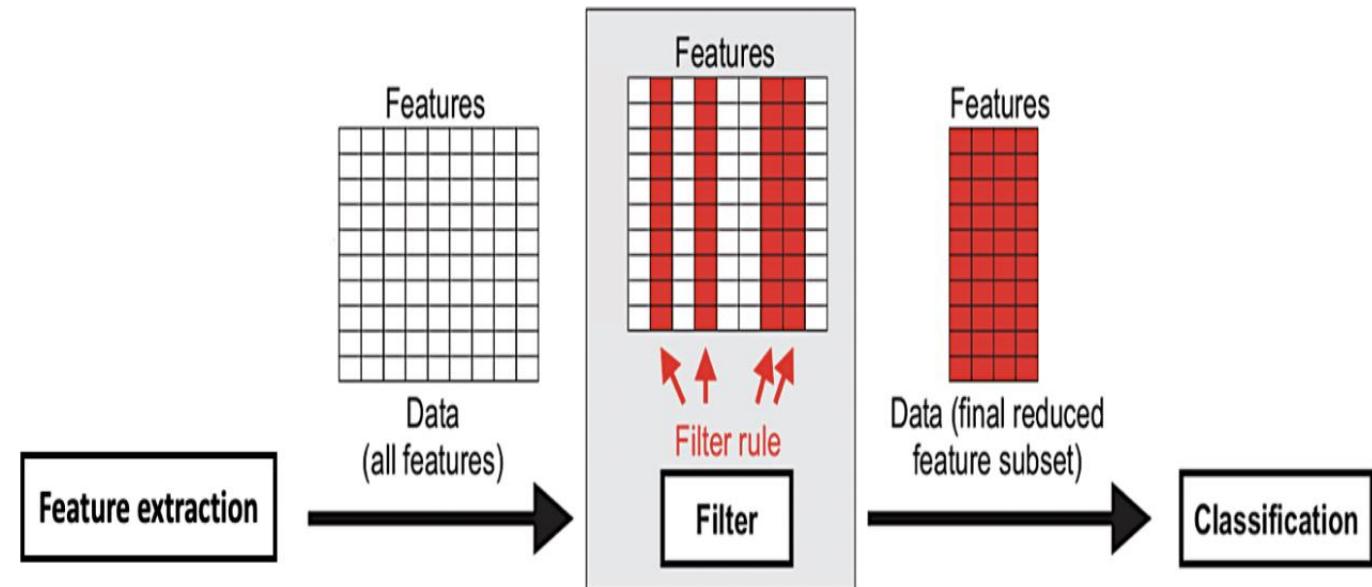
- The distributions and the relationships among the features can have a significant effect on the performance of the translation algorithm that follows feature extraction. These effects depend on the characteristics of the particular translation algorithm.
 - ***NORMALIZATION***
 - ***LOG-NORMAL TRANSFORMS***
 - ***FEATURE SMOOTHING***

Feature selection

- Feature selection methods helps to select or drop features depending on some performance measure.
- Reducing the number of irrelevant features will drastically improve the learning performance, lower the computational complexity and decrease the required storage.
- The most commonly used feature selection strategies are forward search methods and backward elimination methods. The former method starts with one feature initially and builds large feature set iteratively, whereas the latter starts with the large feature set and remove features iteratively.
- Filter method, wrapper method and embedded method

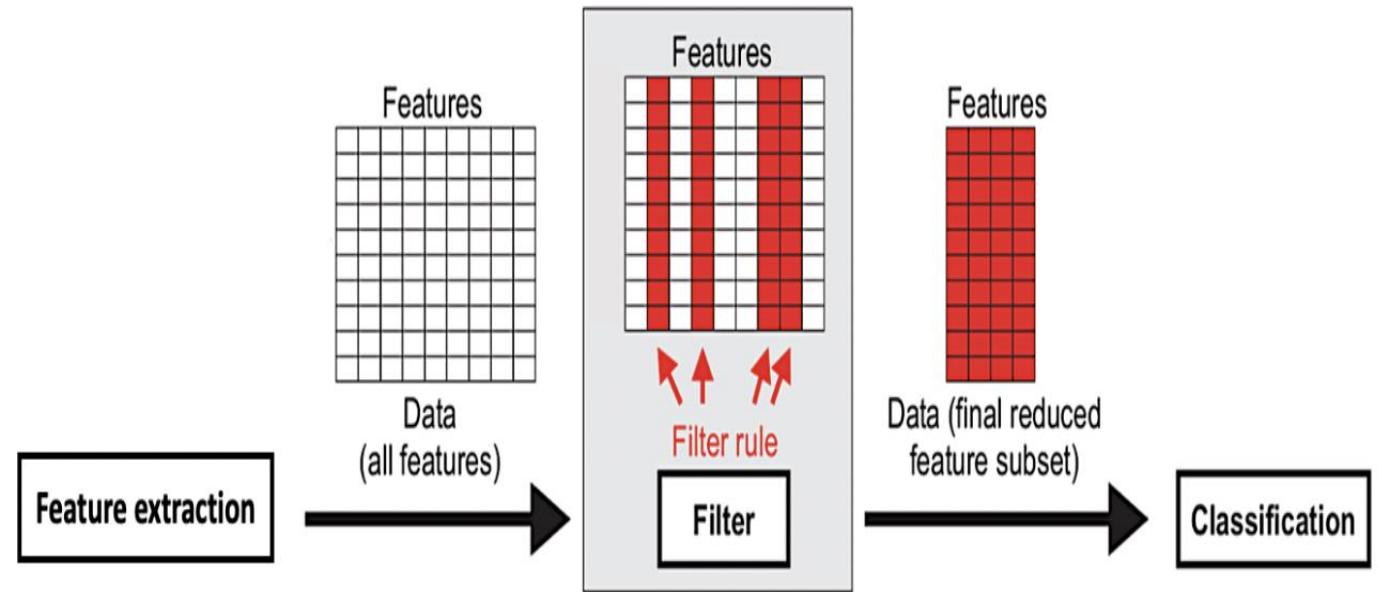
Feature selection – Filter Method

- Filter methods chooses a feature subset from the large feature set applying some filter rules before training the classifier model.
- The filter rule may be derived from prior knowledge or using some data statistics.
- In order to work efficiently, filter methods need a strong assumptions about the class distribution.
- If the rightful assumptions are given, they work very fast and return best possible solutions.
- On the other hand, if the false assumptions are considered, filter methods perform very poor.



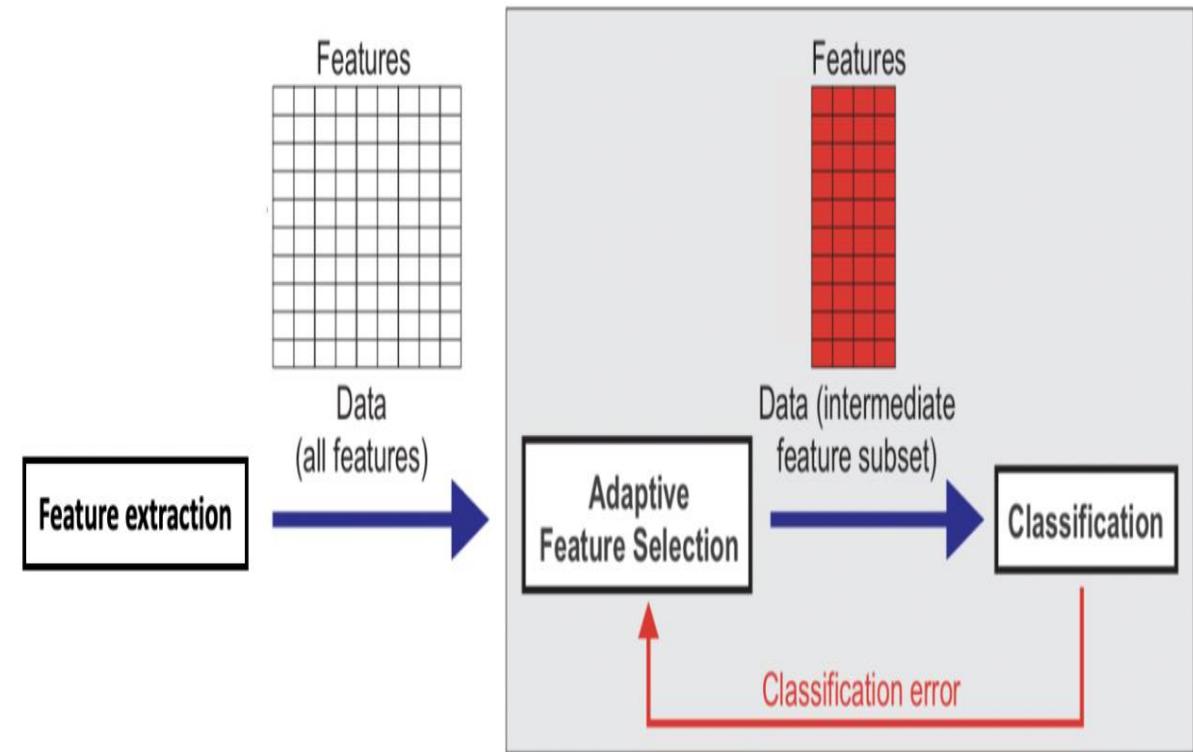
Feature selection – Filter Method

- Filter methods rank the features one-by-one based on the relevance score.
- Finally, the user need to define the number of features or a threshold of a score to determine the subset of features.
- The methods that follows filter approach are correlation coefficient, mutual information, fisher criterion, Laplacian score, etc.



Feature selection – Wrapper Method

- In wrapper method, the process of feature selection takes place in an iterative aspect by tuning the feature subset to a classifier.
- The chosen feature subset is entirely dependent on the evaluated classification error. The feature subset with a very small estimated classification error is preferred.
- The wrapper approach is considered to be expensive as it depends on the classification error at every iteration.
- The popular search strategies that follows wrapper approach are sequential forward selection (SFS), sequential backward elimination (SBE), sequential floating forward selection (SFFS), Genetic algorithms, etc.



Feature selection – Embedded Method

- An embedded method is a combination of filter method and wrapper method.
- It takes advantages of both the methods to avoid the manual specification of threshold score and at the same time, learns which feature subset contribute to the best accuracy of the classifier model.
- The common embedded feature selection methods are regularization methods which introduces constraints into the optimization of a predictive algorithm.
- Examples of embedded feature selection methods are LASSO (least absolute shrinkage and selection operator), elastic net, etc.
- The computational complexity of this approach tends to be between filter and wrapper methods.

Thank You!



Brain Computer Interaction

Feature Translation

Course Instructors

Dr. Sreeja S R

Assistant Professor

Indian Institute of Information Technology
IIIT Sri City

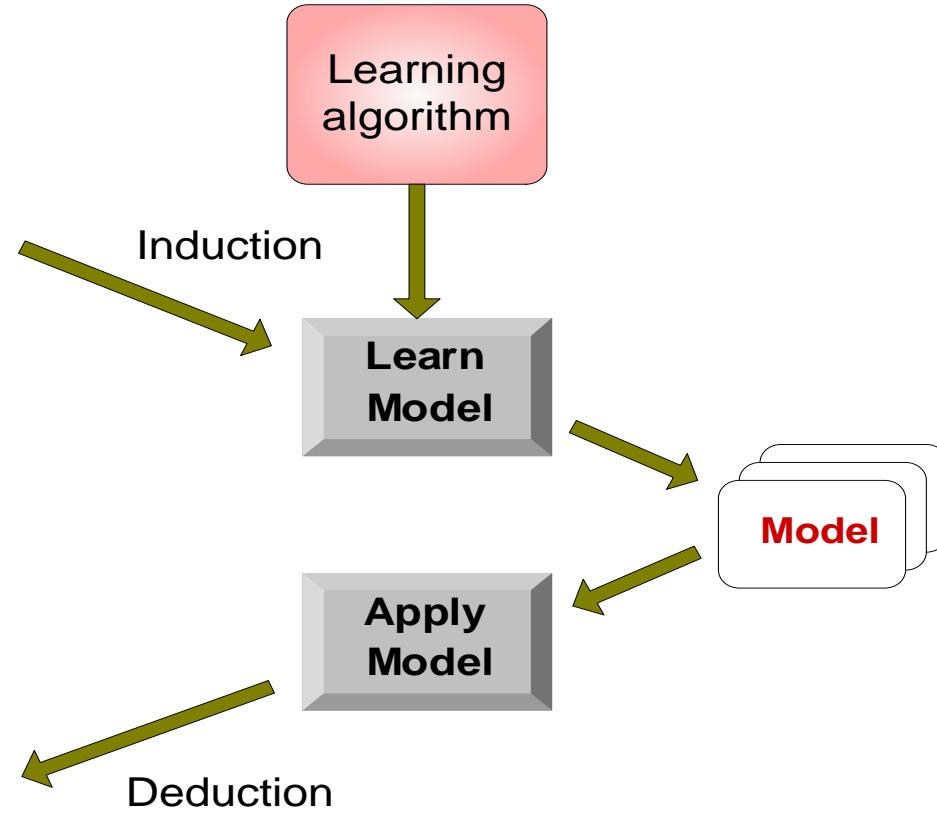
Illustrating Classification Tasks

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

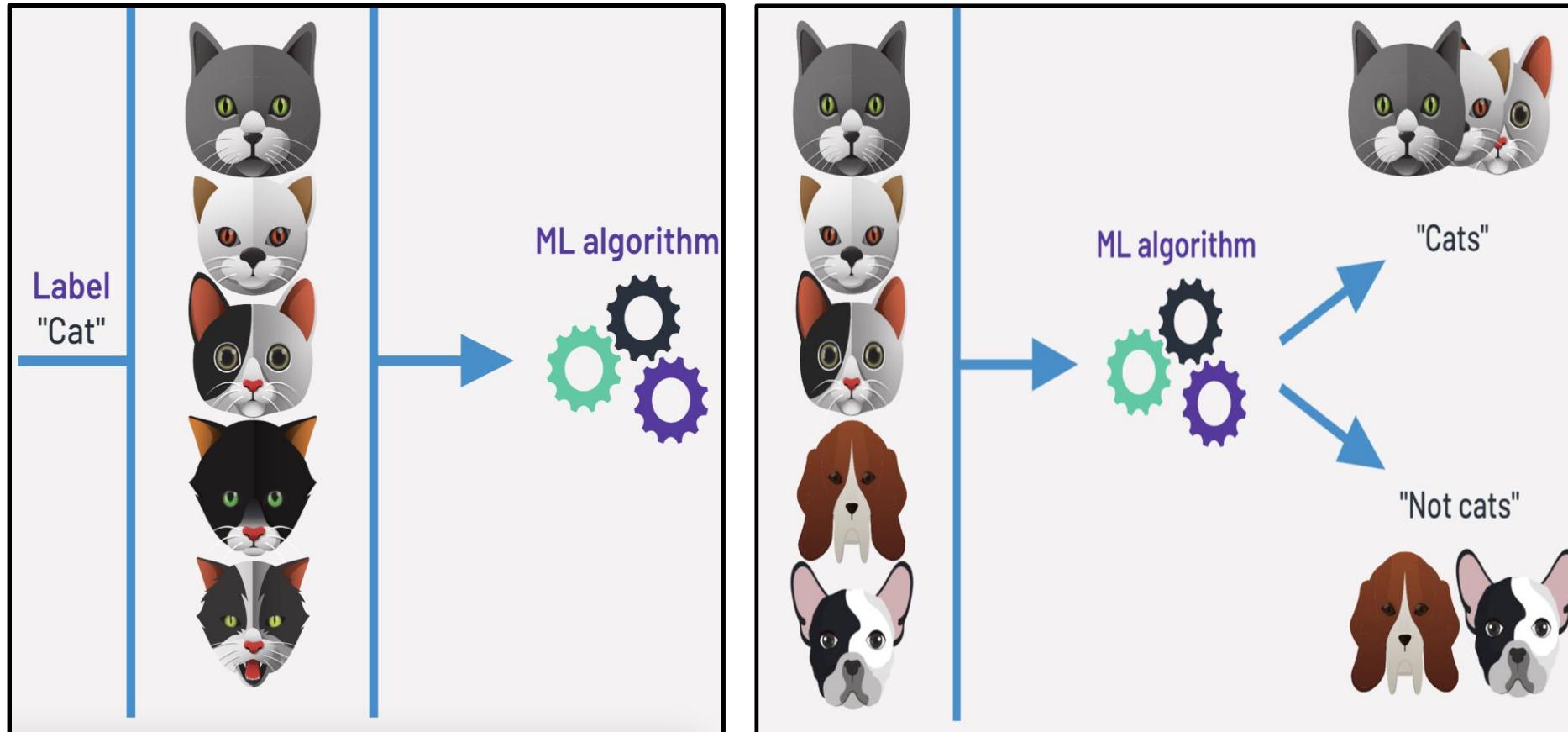
Test Set



Classification Techniques

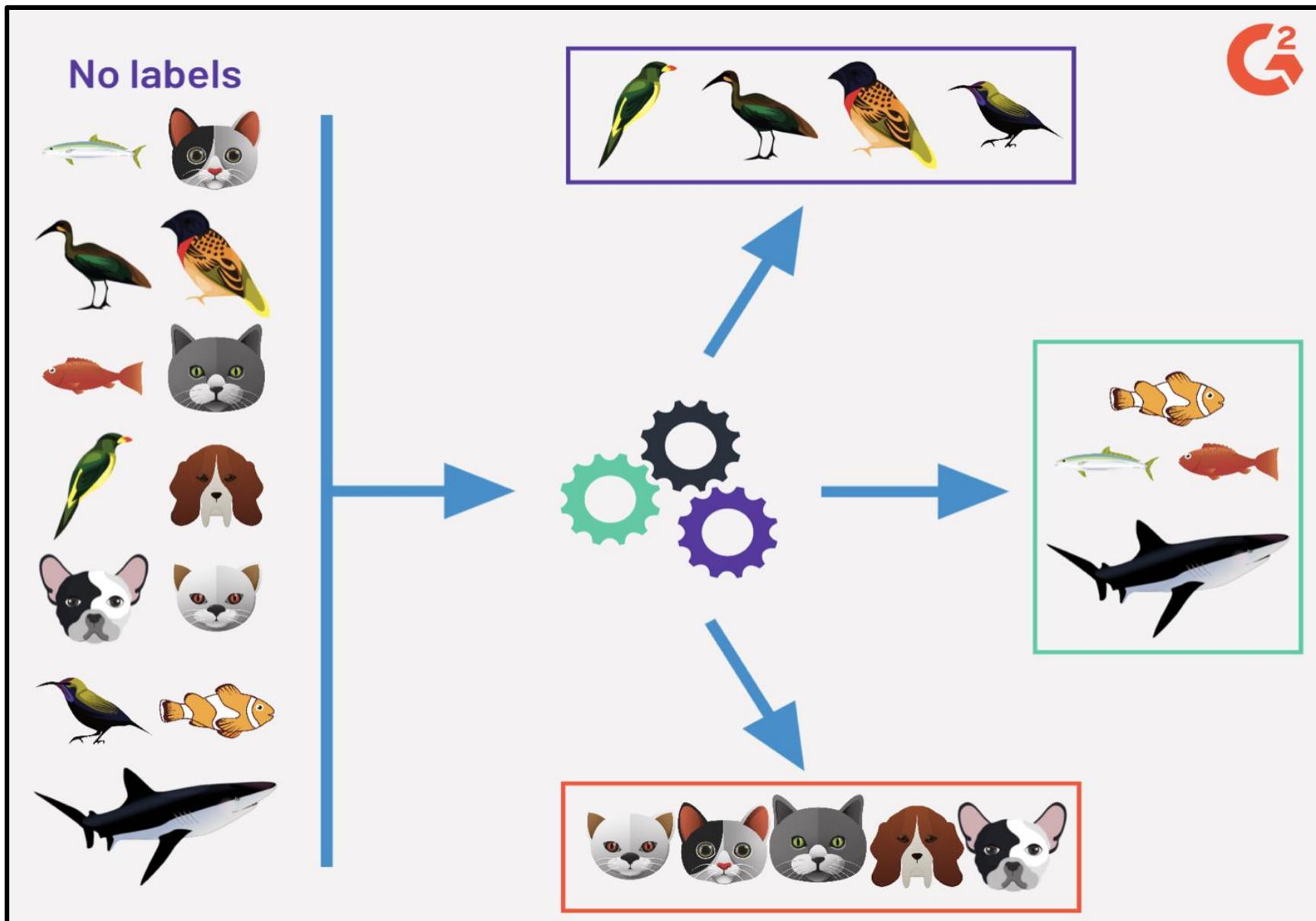
- Classification consists of assigning a class label to a set of unclassified cases.
- **Supervised Classification**
 - The set of possible classes is known in advance.
- **Unsupervised Classification**
 - Set of possible classes is not known. After classification we can try to assign a name to that class.
 - Unsupervised classification is called **clustering**.

Supervised Classification



Source: <https://www.g2.com/articles/supervised-vs-unsupervised-learning>

Unsupervised Classification

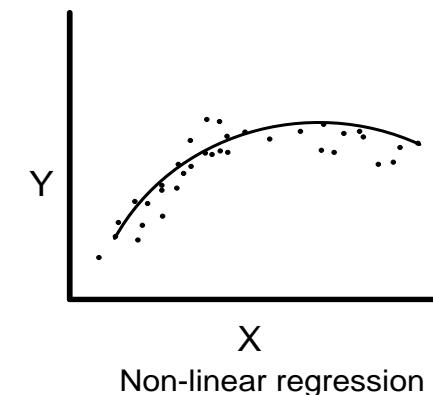
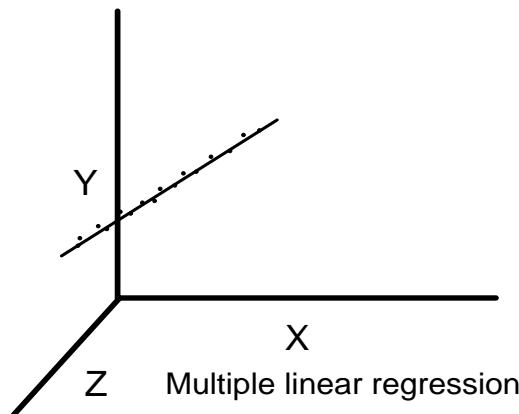
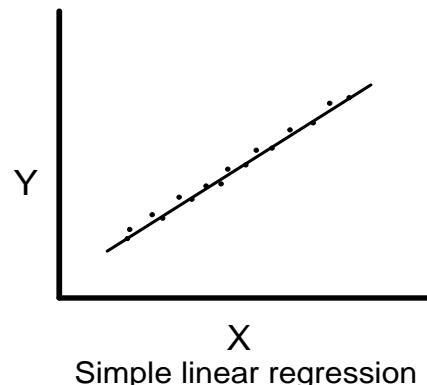


Supervised Classification Technique

- Given a collection of records (*training set*)
 - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: *Previously unseen records* should be assigned a class as accurately as possible.
 - Satisfy the property of “mutually exclusive and exhaustive”

Regression Analysis

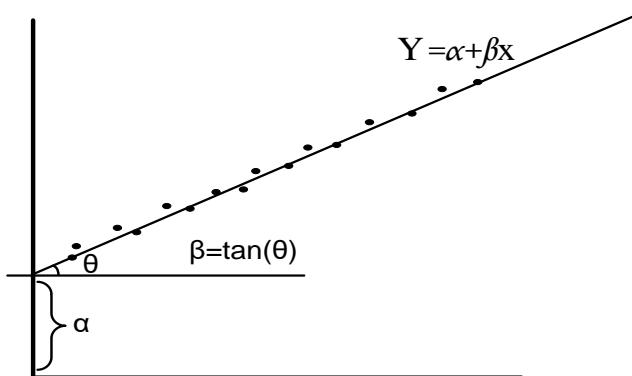
- The regression analysis is a statistical method to deal with the formulation of mathematical model depicting relationship amongst variables, which can be used for the purpose of prediction of the values of dependent variable, given the values of independent variables.
- Classification of Regression Analysis Models**
 - Linear regression models
 - Simple linear regression
 - Multiple linear regression
 - Non-linear regression models



Simple Linear Regression Model

In simple linear regression, we have only two variables:

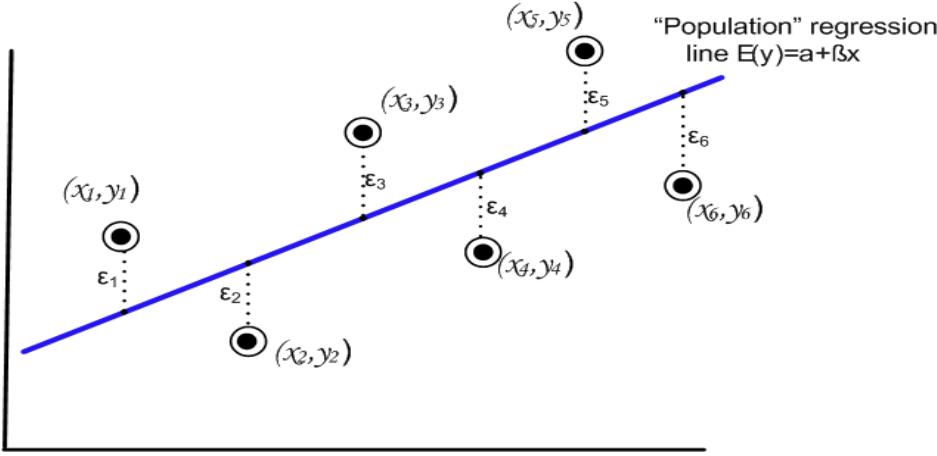
- Dependent variable (also called **Response**), usually denoted as Y .
- Independent variable (alternatively called **Regressor**), usually denoted as x .
- A reasonable form of a relationship between the Response Y and the Regressor x is the linear relationship, that is in the form $Y = \alpha + \beta x$



Note:

- There are infinite number of lines (and hence α_s and β_s)
- The concept of regression analysis deal with finding the best relationship between Y and x (and hence best fitted values of α and β) quantifying the strength of that relationship.

Regression Analysis



Given the set $[(x_i, y_i), i = 1, 2, \dots, n]$ of data involving n pairs of (x, y) values, our objective is to find “true” or population regression line such that $Y = \alpha + \beta x + \epsilon$

Here, ϵ is a random variable with $E(\epsilon) = 0$ and $var(\epsilon) = \sigma^2$. The quantity σ^2 is often called the **error variance**.

Note:

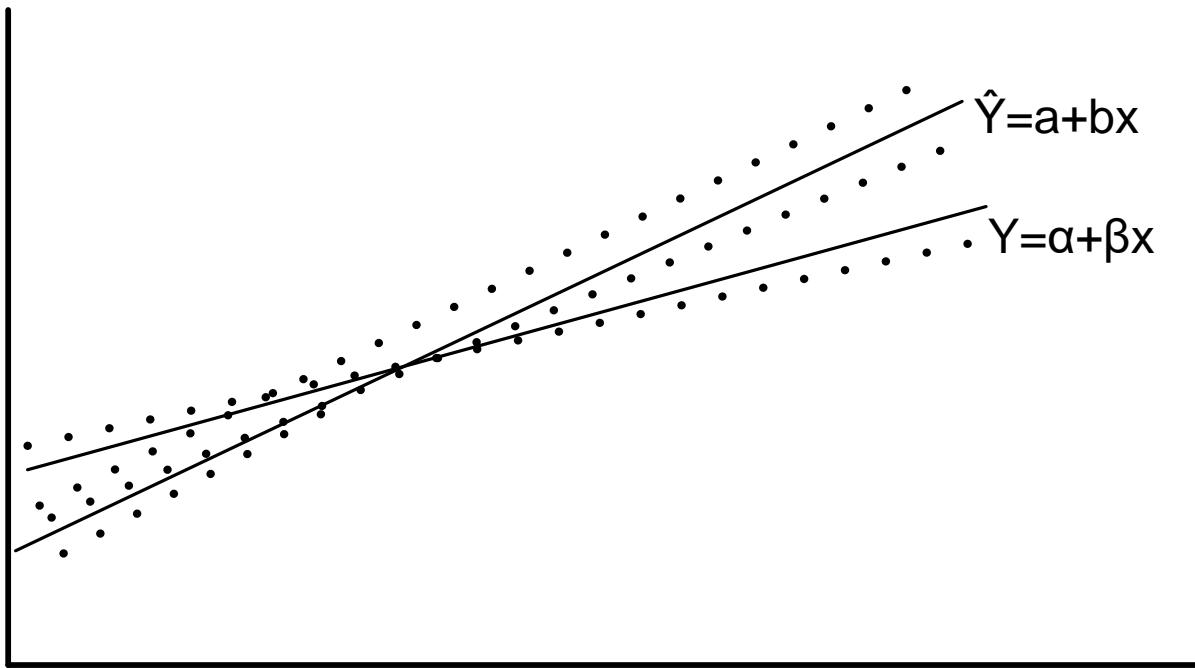
- $E(\epsilon) = 0$ implies that at a specific x , the y values are distributed around the “true” regression line $Y = \alpha + \beta x$ (i.e., the positive and negative errors around the true line is reasonable).
- α and β are called **regression coefficients**.
- α and β values are to be estimated from the data.

True versus Fitted Regression Line

- The task in regression analysis is to estimate the regression coefficients α and β .
- Suppose, we denote the estimates a for α and b for β . Then the fitted regression line is

$$\hat{Y} = a + bx$$

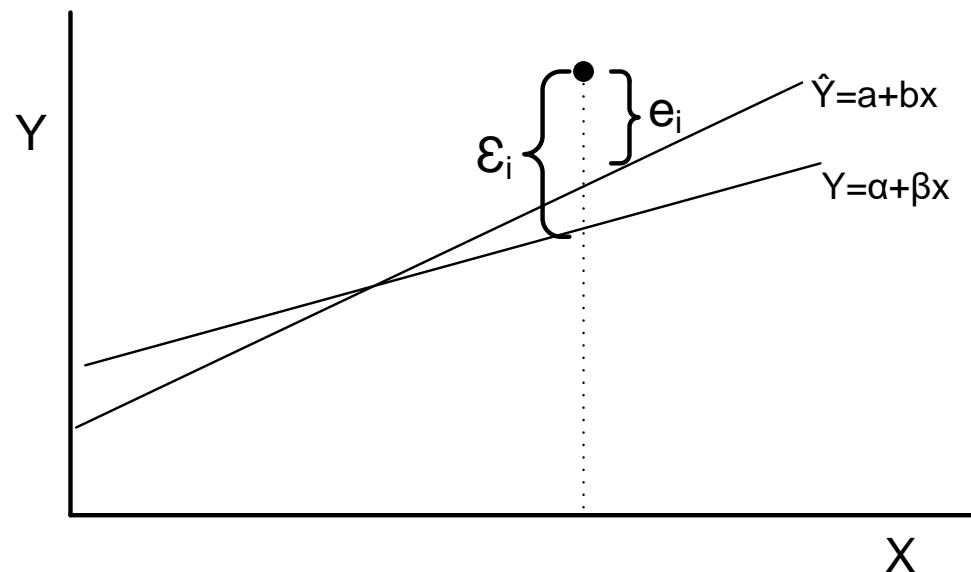
where \hat{Y} is the predicted or fitted value.



Least Square Method to estimate α and β

This method uses the concept of **residual**. A residual is essentially an error in the fit of the model $\hat{Y} = a + bx$. Thus, i^{th} residual is

$$e_i = Y_i - \hat{Y}_i, i = 1, 2, 3, \dots, n$$



Least Square method

- The residual sum of squares is often called **the sum of squares of the errors** about the fitted line and is denoted as SSE

$$SSE = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - a - bx_i)^2$$

- We are to minimize the value of SSE and hence to determine the parameters of a and b .
- Differentiating SSE with respect to a and b , we have

$$\frac{\partial(SSE)}{\partial a} = -2 \sum_{i=1}^n (y_i - a - bx_i)$$

$$\frac{\partial(SSE)}{\partial b} = -2 \sum_{i=1}^n (y_i - a - bx_i) \cdot x_i$$

For minimum value of SSE, $\frac{\partial(SSE)}{\partial a} = 0$

$$\frac{\partial(SSE)}{\partial b} = 0$$

Least Square method to estimate α and β

Thus we set

$$na + b \sum_{i=1}^n x_i = \sum_{i=1}^n y_i$$

$$a \sum_{i=1}^n x_i + b \sum_{i=1}^n x_i^2 = \sum_{i=1}^n x_i y_i$$

These two equations can be solved to determine the values of a and b , and it can be calculated that

$$b = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$a = \bar{y} - b\bar{x}$$

R^2 : Measure of Quality of Fit

- A quantity R^2 , is called **coefficient of determination** is used to measure the proportion of variability of the fitted model.
- We have $SSE = \sum_{i=1}^n (y_i - \hat{y})^2$
- It signifies the **variability due to error**.
- Now, let us define the **total corrected sum of squares**, defined as

$$SST = \sum_{i=1}^n (y_i - \bar{y})^2$$

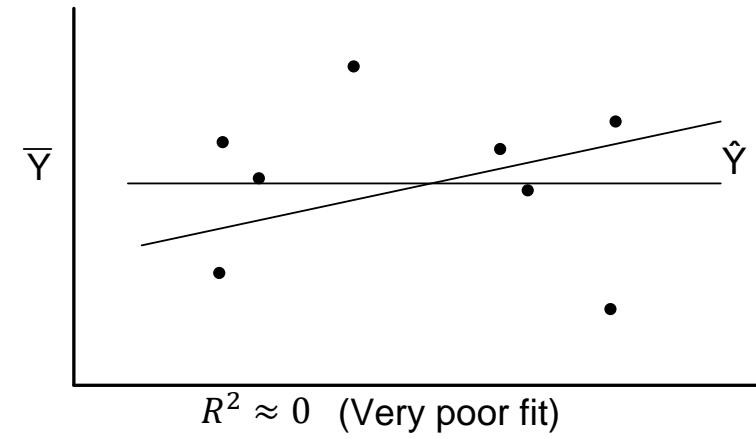
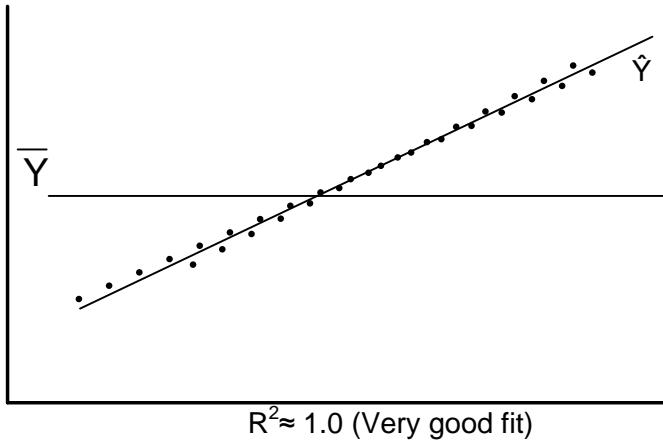
- SST represents the variation in the response values. The R^2 is

$$R^2 = 1 - \frac{SSE}{SST}$$

Note:

- If fit is perfect, all residuals are zero and thus $R^2 = 1.0$ (very good fit)
- If SSE is only slightly smaller than SST, then $R^2 \approx 0$ (very poor fit)

R^2 : Measure of Quality of Fit



Any question?

Thank You!



Brain Computer Interaction

Feature Translation

Course Instructors

Dr. Sreeja S R

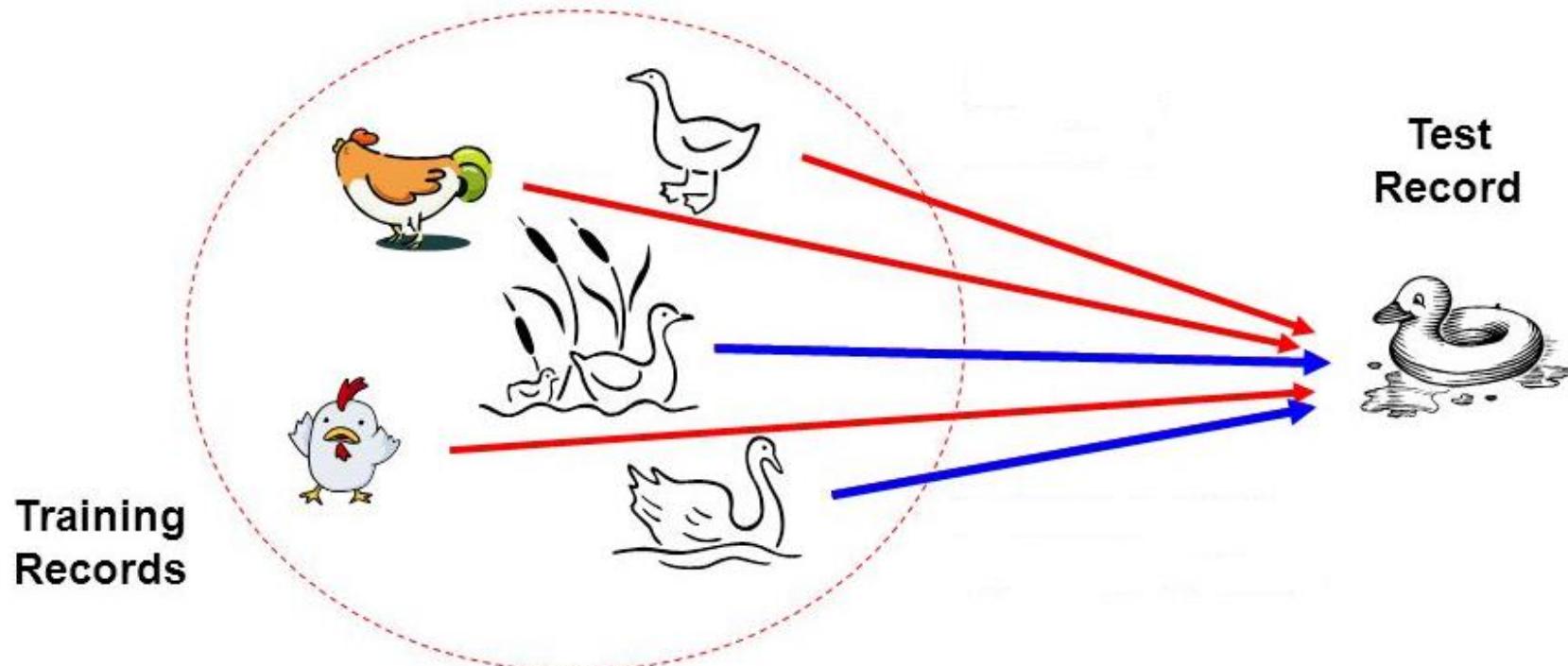
Assistant Professor

Indian Institute of Information Technology
IIIT Sri City

Bayesian Classifier

Bayesian Classifier

- Principle
 - If it walks like a duck, quacks like a duck, then it is **probably** a duck

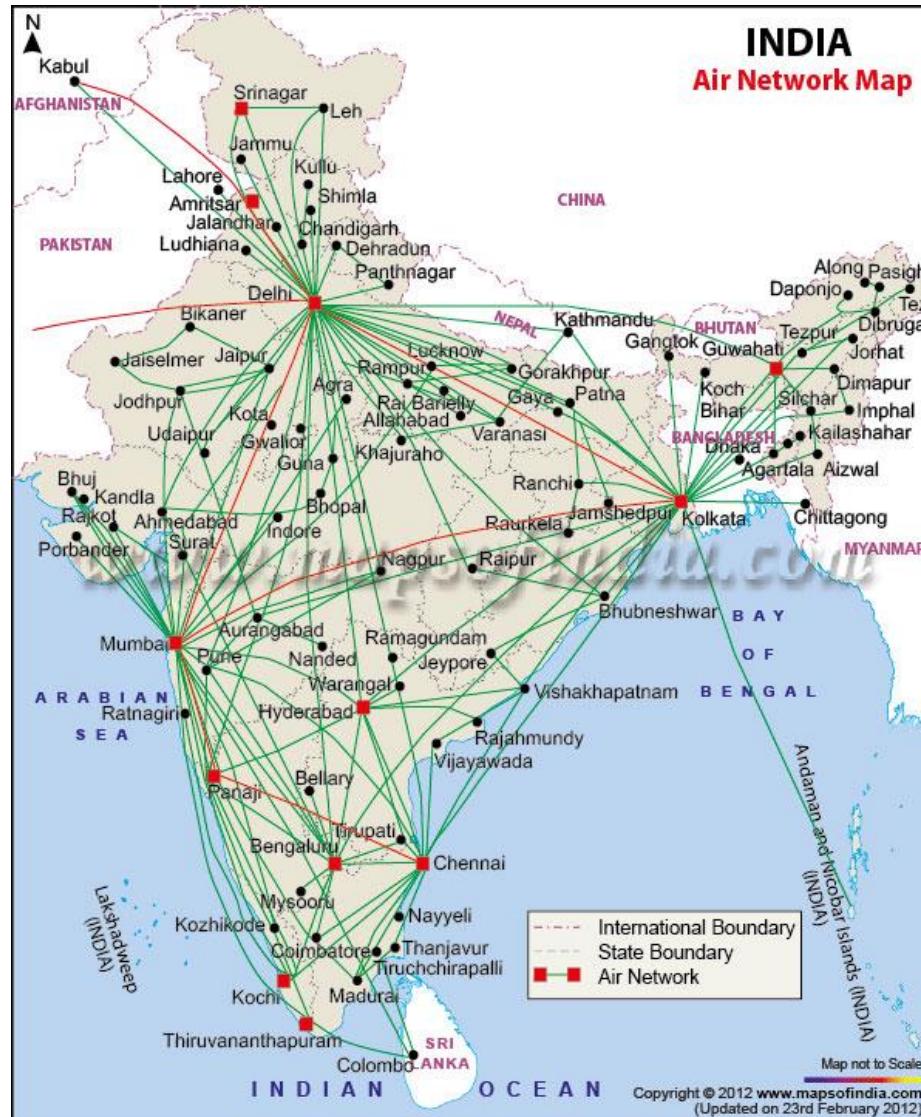


Bayesian Classifier

- A statistical classifier
 - Performs *probabilistic prediction*, i.e., predicts class membership probabilities
- Foundation
 - Based on Bayes' Theorem.
- Assumptions
 1. The classes are mutually exclusive and exhaustive.
 2. The attributes are independent given the class.
- Called “Naïve” classifier because of these assumptions.
 - Empirically proven to be useful.
 - Scales very well.

Example: Bayesian Classification

- Example : Air Traffic Data
 - Let us consider a set observation recorded in a database
 - Regarding the arrival of airplanes in the routes from any airport to New Delhi under certain conditions.



Air-Traffic Data

Days	Season	Fog	Rain	Class
Weekday	Spring	None	None	On Time
Weekday	Winter	None	Slight	On Time
Weekday	Winter	None	None	On Time
Holiday	Winter	High	Slight	Late
Saturday	Summer	Normal	None	On Time
Weekday	Autumn	Normal	None	Very Late
Holiday	Summer	High	Slight	On Time
Sunday	Summer	Normal	None	On Time
Weekday	Winter	High	Heavy	Very Late
Weekday	Summer	None	Slight	On Time

Cond. to next slide...

Air-Traffic Data

Cond. from previous slide...

Days	Season	Fog	Rain	Class
Saturday	Spring	High	Heavy	Cancelled
Weekday	Summer	High	Slight	On Time
Weekday	Winter	Normal	None	Late
Weekday	Summer	High	None	On Time
Weekday	Winter	Normal	Heavy	Very Late
Saturday	Autumn	High	Slight	On Time
Weekday	Autumn	None	Heavy	On Time
Holiday	Spring	Normal	Slight	On Time
Weekday	Spring	Normal	None	On Time
Weekday	Spring	Normal	Heavy	On Time

Air-Traffic Data

- In this database, there are four attributes

$$A = [\text{Day}, \text{Season}, \text{Fog}, \text{Rain}]$$

with 20 tuples.

- The categories of classes are:

$$C = [\text{On Time}, \text{Late}, \text{Very Late}, \text{Cancelled}]$$

- Given this is the knowledge of data and classes, we are to find most likely classification for any other [unseen instance](#), for example:

Week Day	Winter	High	None	???
----------	--------	------	------	-----

- Classification technique eventually map this tuple into an accurate class.

Bayesian Classifier

- In many applications, the relationship between the attributes set and the class variable is **non-deterministic**.
 - In other words, a test cannot be classified to a class label with certainty.
 - In such a situation, the classification can be achieved **probabilistically**.
- The Bayesian classifier is an approach for **modelling probabilistic relationships** between the attribute set and the class variable.
- More precisely, Bayesian classifier use **Bayes' Theorem of Probability** for classification.
- Before going to discuss the Bayesian classifier, we should have a quick look at the **Theory of Probability** and then **Bayes' Theorem**.

Bayes' Theorem of Probability

Simple Probability

Definition 8.2: Simple Probability

If there are n elementary events associated with a random experiment and m of n of them are favorable to an event A , then the probability of happening or occurrence of A is

$$P(A) = \frac{m}{n}$$

Simple Probability

- Suppose, A and B are any two events and $P(A)$, $P(B)$ denote the probabilities that the events A and B will occur, respectively.
- **Mutually Exclusive Events:**
 - Two events are mutually exclusive, if the occurrence of one precludes the occurrence of the other.

Example: Tossing a coin (two events)

Tossing a ludo cube (Six events)

💡 Can you give an example, so that two events are not mutually exclusive?

Hint: Tossing two identical coins, Heart and King

Simple Probability

- **Independent events:** Two events are independent if occurrences of one does not alter the occurrence of other.

Example: Tossing both coin and ludo cube together.

💡 Can you give an example, where an event is dependent on one or more other events(s)?

Hint: Receiving a message (A) through a communication channel (B) over a computer (C), rain and train.

Joint Probability

Definition 8.3: Joint Probability

If $P(A)$ and $P(B)$ are the probability of two events, then

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

If A and B are mutually exclusive, then $P(A \cap B) = 0$

If A and B are independent events, then $P(A \cap B) = P(A).P(B)$

Thus, for mutually exclusive events

$$P(A \cup B) = P(A) + P(B)$$

Conditional Probability

Definition 8.2: Conditional Probability

If events are dependent, then their probability is expressed by conditional probability. The probability that A occurs given that B is denoted by $P(A|B)$.

Suppose, A and B are two events associated with a random experiment. The probability of A under the condition that B has already occurred and $P(B) \neq 0$ is given by

$$\begin{aligned} P(A|B) &= \frac{\text{Number of events in } B \text{ which are favourable to } A}{\text{Number of events in } B} \\ &= \frac{\text{Number of events favourable to } A \cap B}{\text{Number of events favourable to } B} \\ &= \frac{P(A \cap B)}{P(B)} \end{aligned}$$

Conditional Probability

Corollary 8.1: Conditional Probability

$$P(A \cap B) = P(A) \cdot P(B|A), \quad \text{if } P(A) \neq 0$$

or $P(A \cap B) = P(B) \cdot P(A|B), \quad \text{if } P(B) \neq 0$

For three events A, B and C

$$P(A \cap B \cap C) = P(A) \cdot P(B) \cdot P(C|A \cap B)$$

For n events A_1, A_2, \dots, A_n and if all events are mutually independent to each other

$$P(A_1 \cap A_2 \cap \dots \cap A_n) = P(A_1) \cdot P(A_2) \dots \dots \dots P(A_n)$$

Note:

$P(A|B) = 0$ if events are **mutually exclusive**

$P(A|B) = P(A)$ if A and B are **independent**

$P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$ otherwise,

$P(A \cap B) = P(B \cap A)$

Conditional Probability

- Generalization of Conditional Probability:

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

$$= \frac{P(B|A) \cdot P(A)}{P(B)} \quad \because P(A \cap B) = P(B|A) \cdot P(A) = P(A|B) \cdot P(B)$$

By the law of total probability : $P(B) = P[(B \cap A) \cup (B \cap \bar{A})]$, where \bar{A} denotes the compliment of event A. Thus,

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P[(B \cap A) \cup (B \cap \bar{A})]}$$

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

Conditional Probability

In general,

$$P(A|D) = \frac{P(A) \cdot P(D|A)}{P(A) \cdot P(D|A) + P(B) \cdot P(D|B) + P(C) \cdot P(D|C)}$$

Total Probability

Definition 8.3: Total Probability

Let E_1, E_2, \dots, E_n be n mutually exclusive and exhaustive events associated with a random experiment. If A is any event which occurs with E_1 or E_2 or \dots, E_n , then

$$P(A) = P(E_1).P(A|E_1) + P(E_2).P(A|E_2) + \dots + P(E_n).P(A|E_n)$$

Bayes' Theorem

Theorem 8.4: Bayes' Theorem

Let E_1, E_2, \dots, E_n be n mutually exclusive and exhaustive events associated with a random experiment. If A is any event which occurs with E_1 or E_2 or ... or E_n , then

$$P(E_i|A) = \frac{P(E_i) \cdot P(A|E_i)}{\sum_{i=1}^n P(E_i) \cdot P(A|E_i)}$$

Prior and Posterior Probabilities

- $P(A)$ and $P(B)$ are called prior probabilities
- $P(A|B)$, $P(B|A)$ are called posterior probabilities

Example 8.6:

This table shows that the event Y has two outcomes namely A and B , which is dependent on another event X with various outcomes like x_1 , x_2 and x_3 .

- **Case1:** Suppose, we don't have any information of the event A . Then, from the given sample space, we can calculate $P(Y = A) = \frac{5}{10} = 0.5$
- **Case2:** Now, suppose, we want to calculate $P(X = x_2 | Y = A) = \frac{2}{5} = 0.4$.

The later is the likelihood, whereas the former is the prior probability.

X	Y
x_1	A
x_2	A
x_3	B
x_3	A
x_2	B
x_1	A
x_1	B
x_3	B
x_2	B
x_2	A

Naïve Bayesian Classifier

- Suppose, Y is a class variable and $X = \{X_1, X_2, \dots, X_n\}$ is a set of attributes, with instance of Y .

INPUT (X)	CLASS(Y)
...	...
...	...
x_1, x_2, \dots, x_n	y_i
...	...

- The classification problem, then can be expressed as the class-conditional probability

$$P(Y = y_i | (X_1 = x_1) \text{ AND } (X_2 = x_2) \text{ AND } \dots \text{ AND } (X_n = x_n))$$

Naïve Bayesian Classifier

- Naïve Bayesian classifier calculate this posterior probability using Bayes' theorem, which is as follows.
- From Bayes' theorem on conditional probability, we have

$$\begin{aligned} P(Y|X) &= \frac{P(X|Y) \cdot P(Y)}{P(X)} \\ &= \frac{P(X|Y) \cdot P(Y)}{P(X|Y = y_1) \cdot P(Y = y_1) + \cdots + P(X|Y = y_k) \cdot P(Y = y_k)} \end{aligned}$$

where,

$$P(X) = \sum_{i=1}^k P(X|Y = y_i) \cdot P(Y = y_i)$$

Note:

- $P(X)$ is called the evidence (also the total probability) and it is a constant.
- The probability $P(Y|X)$ (also called class conditional probability) is therefore proportional to $P(X|Y) \cdot P(Y)$.
- Thus, $P(Y|X)$ can be taken as a measure of Y given that X .

$$P(Y|X) \approx P(X|Y) \cdot P(Y)$$

Naïve Bayesian Classifier

- Suppose, for a given instance of X (say $x = (X_1 = x_1)$ and $(X_n = x_n)$).
- There are any two class conditional probabilities namely $P(Y = y_i | X=x)$ and $P(Y = y_j | X=x)$.
- If $P(Y = y_i | X=x) > P(Y = y_j | X=x)$, then we say that y_i is more stronger than y_j for the instance $X = x$.
- The strongest y_i is the classification for the instance $X = x$.

Naïve Bayesian Classifier

- **Example:** With reference to the Air Traffic Dataset mentioned earlier, let us tabulate all the posterior and prior probabilities as shown below.

		Class			
Attribute		On Time	Late	Very Late	Cancelled
Day	Weekday	9/14 = 0.64	½ = 0.5	3/3 = 1	0/1 = 0
	Saturday	2/14 = 0.14	½ = 0.5	0/3 = 0	1/1 = 1
	Sunday	1/14 = 0.07	0/2 = 0	0/3 = 0	0/1 = 0
	Holiday	2/14 = 0.14	0/2 = 0	0/3 = 0	0/1 = 0
Season	Spring	4/14 = 0.29	0/2 = 0	0/3 = 0	0/1 = 0
	Summer	6/14 = 0.43	0/2 = 0	0/3 = 0	0/1 = 0
	Autumn	2/14 = 0.14	0/2 = 0	1/3 = 0.33	0/1 = 0
	Winter	2/14 = 0.14	2/2 = 1	2/3 = 0.67	0/1 = 0

Naïve Bayesian Classifier

		Class			
Attribute		On Time	Late	Very Late	Cancelled
Fog	None	5/14 = 0.36	0/2 = 0	0/3 = 0	0/1 = 0
	High	4/14 = 0.29	1/2 = 0.5	1/3 = 0.33	1/1 = 1
	Normal	5/14 = 0.36	1/2 = 0.5	2/3 = 0.67	0/1 = 0
Rain	None	5/14 = 0.36	1/2 = 0.5	1/3 = 0.33	0/1 = 0
	Slight	8/14 = 0.57	0/2 = 0	0/3 = 0	0/1 = 0
	Heavy	1/14 = 0.07	1/2 = 0.5	2/3 = 0.67	1/1 = 1
Prior Probability		14/20 = 0.70	2/20 = 0.10	3/20 = 0.15	1/20 = 0.05

Naïve Bayesian Classifier

Instance:

Week Day	Winter	High	None	???
----------	--------	------	------	-----

Case1: Class = On Time : $0.70 \times 0.64 \times 0.14 \times 0.29 \times 0.36 = 0.0065$

Case2: Class = Late : $0.10 \times 0.50 \times 1.0 \times 0.50 \times 0.50 = 0.0125$

Case3: Class = Very Late : $0.15 \times 1.0 \times 0.67 \times 0.33 \times 0.33 = 0.0109$

Case4: Class = Cancelled : $0.05 \times 0.0 \times 0.0 \times 1.0 \times 0.0 = 0.0000$

Case2 is the strongest; Hence correct classification is **Late**

Naïve Bayesian Classifier

Algorithm: Naïve Bayesian Classification

Input: Given a set of k mutually exclusive and exhaustive classes $C = \{c_1, c_2, \dots, c_k\}$, which have prior probabilities $P(C_1), P(C_2), \dots, P(C_k)$.

There are n -attribute set $A = \{A_1, A_2, \dots, A_n\}$, which for a given instance have values $A_1 = a_1, A_2 = a_2, \dots, A_n = a_n$

Step: For each $c_i \in C$, calculate the class condition probabilities, $i = 1, 2, \dots, k$

$$p_i = P(C_i) \times \prod_{j=1}^n P(A_j = a_j | C_i)$$

$$p_x = \max\{p_1, p_2, \dots, p_k\}$$

Output: C_x is the classification

Note: $\sum p_i \neq 1$, because they are not probabilities rather proportion values (to posterior probabilities)

Naïve Bayesian Classifier

Pros and Cons

- The Naïve Bayes' approach is a very popular one, which often works well.
- However, it has a number of potential problems
 - It relies on all attributes being **categorical**.
 - If the data is **less**, then it **estimates poorly**.

Naïve Bayesian Classifier

Approach to overcome the limitations in Naïve Bayesian Classification

- Estimating the posterior probabilities for continuous attributes
 - In real life situation, all attributes are not necessarily be categorical, In fact, there is a mix of both categorical and continuous attributes.
 - In the following, we discuss the schemes to deal with continuous attributes in Bayesian classifier.
 1. We can discretize each continuous attributes and then replace the continuous values with its corresponding discrete intervals.
 2. We can assume a certain form of probability distribution for the continuous variable and estimate the parameters of the distribution using the training data. A Gaussian distribution is usually chosen to represent the posterior probabilities for continuous attributes. A general form of Gaussian distribution will look like

$$P(x: \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x - \mu)^2}{2\sigma^2}}$$

where, μ and σ^2 denote mean and variance, respectively.

Naïve Bayesian Classifier

For each class C_i , the posterior probabilities for attribute A_j (it is the numeric attribute) can be calculated following Gaussian normal distribution as follows.

$$P(A_j = a_j | C_i) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} e^{-\frac{(a_j - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

Here, the parameter μ_{ij} can be calculated based on the sample mean of attribute value of A_j for the training records that belong to the class C_i .

Similarly, σ_{ij}^2 can be estimated from the calculation of variance of such training records.

Naïve Bayesian Classifier

M-estimate of Conditional Probability

- The M-estimation is to deal with the potential problem of Naïve Bayesian Classifier when training data size is too poor.
 - If the likelihood probability for one of the attribute is zero, then the overall class-conditional probability for the class vanishes.
 - In other words, if training data do not cover many of the attribute values, then we may not be able to classify some of the test records.
- This problem can be addressed by using the M-estimate approach.

M-estimate Approach

- M-estimate approach can be stated as follows

$$P(A_j = a_j | C_i) = \frac{n_{c_i} + mp}{n + m}$$

where, n = total number of instances from class C_i

n_{c_i} = number of training examples from class C_i that take the value $A_j = a_j$

m = it is a parameter known as the equivalent sample size, and

p = is a user specified parameter.

Note:

If $n = 0$, that is, if there is no training set available, then $P(a_i | C_i) = p$,
so, this is a different value, in absence of sample value.

A Practice Example

Example 8.4

Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

Data instance

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

A Practice Example

- $P(C_i)$: $P(\text{buys_computer} = \text{"yes"}) = 9/14 = 0.643$
 $P(\text{buys_computer} = \text{"no"}) = 5/14 = 0.357$
- Compute $P(X|C_i)$ for each class
 - $P(\text{age} = \text{"<=30"} | \text{buys_computer} = \text{"yes"}) = 2/9 = 0.222$
 - $P(\text{age} = \text{"<= 30"} | \text{buys_computer} = \text{"no"}) = 3/5 = 0.6$
 - $P(\text{income} = \text{"medium"} | \text{buys_computer} = \text{"yes"}) = 4/9 = 0.444$
 - $P(\text{income} = \text{"medium"} | \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$
 - $P(\text{student} = \text{"yes"} | \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$
 - $P(\text{student} = \text{"yes"} | \text{buys_computer} = \text{"no"}) = 1/5 = 0.2$
 - $P(\text{credit_rating} = \text{"fair"} | \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$
 - $P(\text{credit_rating} = \text{"fair"} | \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$
- $X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit_rating} = \text{fair})$

$$P(X|C_i) : P(X|\text{buys_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$$
$$P(X|\text{buys_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$$

$$P(X|C_i)*P(C_i) : P(X|\text{buys_computer} = \text{"yes"}) * P(\text{buys_computer} = \text{"yes"}) = 0.028$$
$$P(X|\text{buys_computer} = \text{"no"}) * P(\text{buys_computer} = \text{"no"}) = 0.007$$

Therefore, X belongs to class ("buys_computer = yes")

Any question?

Thank You!



Brain Computer Interaction

Decision Trees

Course Instructors

Dr. Sreeja S R

Assistant Professor

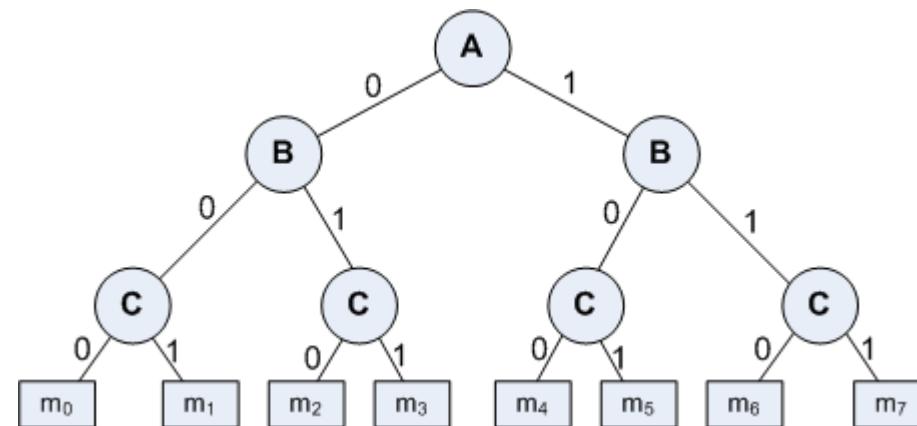
Indian Institute of Information Technology
IIIT Sri City

Basic Concept

- A Decision Tree is an important data structure known to solve many computational problems

Example 8.1: Binary Decision Tree

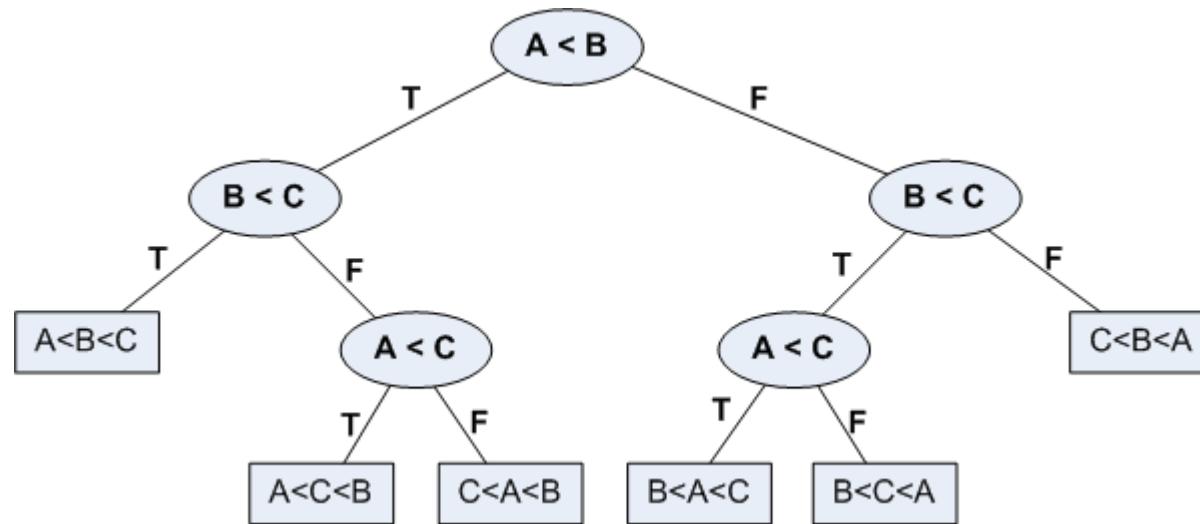
A	B	C	f
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7



Basic Concept

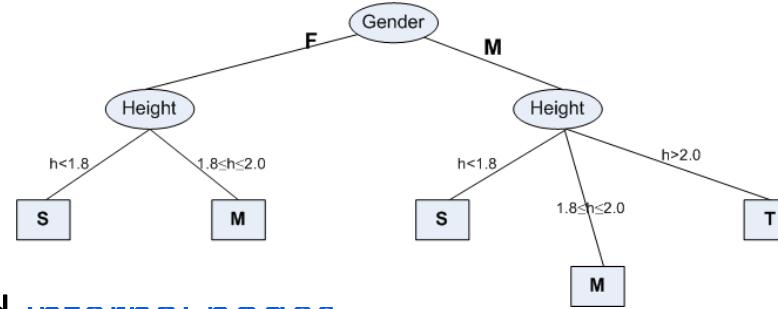
- In Example 18.1, we have considered a decision tree where values of any attribute if binary only. Decision tree is also possible where attributes are of continuous data type

Example 8.2: Decision Tree with numeric data



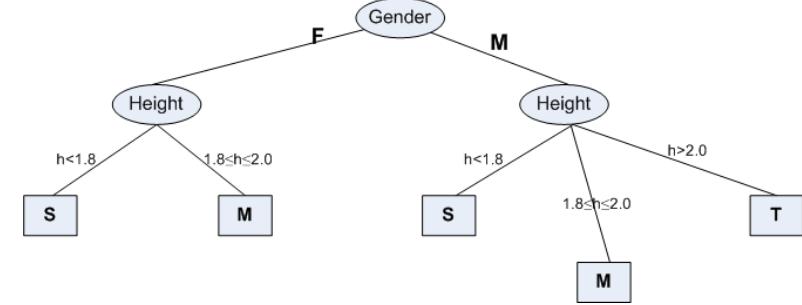
Some Characteristics

- Decision tree may be n -ary, $n \geq 2$.
- There is a special node called **root node**.
- All nodes drawn with circle (ellipse) are called **internal nodes**.
- All nodes drawn with rectangle boxes are called **terminal nodes** or **leaf nodes**.
- Edges of a node represent the **outcome for a value** of the node.
- In a path, a node with same label **is never repeated**.
- Decision tree **is not unique**, as different ordering of internal nodes can give different decision tree.



Decision Tree and Classification Task

- Decision tree helps us to classify data.
 - Internal nodes are some attribute
 - Edges are the values of attributes
 - External nodes are the outcome of classification
- Such a classification is, in fact, made by posing questions starting from the root node to each terminal node.



Decision Tree and Classification Task

Example 8.3 : Vertebrate Classification

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
Human	Warm	hair	yes	no	no	yes	no	Mammal
Python	Cold	scales	no	no	no	no	yes	Reptile
Salmon	Cold	scales	no	yes	no	no	no	Fish
Whale	Warm	hair	yes	yes	no	no	no	Mammal
Frog	Cold	none	no	semi	no	yes	yes	Amphibian
Komodo	Cold	scales	no	no	no	yes	no	Reptile
Bat	Warm	hair	yes	no	yes	yes	yes	Mammal
Pigeon	Warm	feathers	no	no	yes	yes	no	Bird
Cat	Warm	fur	yes	no	no	yes	no	Mammal
Beta	Cold	scales	no	yes	no	no	no	Fish
Turtle	Cold	scales	no	semi	no	yes	no	Reptile
Penguin	Warm	feathers	no	semi	no	yes	no	Bird
Porcupine	Warm	quills	yes	no	no	yes	yes	Mammal
Eel	Cold	scales	no	yes	no	no	no	Fish
Salamander	Cold	none	no	semi	no	yes	yes	Amphibian

What are the class label of Dragon and Shark?

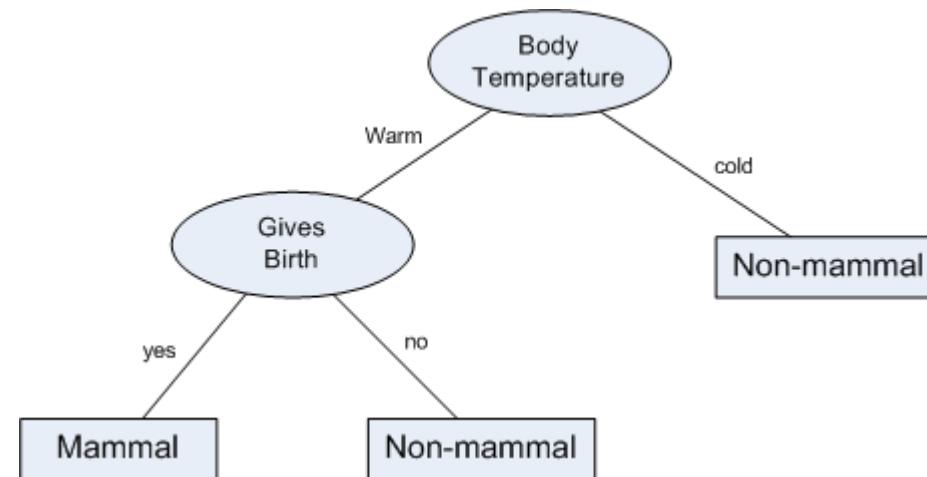
Decision Tree and Classification Task

Example 8.3 : Vertebrate Classification

- Suppose, a new species is discovered as follows.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
Gila Monster	cold	scale	no	no	no	yes	yes	?

- Decision Tree that can be induced based on the data (in Example 8.3) is as follows.



Decision Tree and Classification Task

- The above Example illustrates how we can solve a classification problem by asking a series of question about the attributes.
 - Each time we receive an answer, a follow-up question is asked until we reach a conclusion about the class-label of the test.
- The series of questions and their answers can be organized in the form of a decision tree
 - As a hierarchical structure consisting of nodes and edges
- Once a decision tree is built, it is applied to any test to classify it.

Definition of Decision Tree

Given a database $D = \{t_1, t_2, \dots, t_n\}$, where t_i denotes a tuple, which is defined by a set of attribute $A = \{A_1, A_2, \dots, A_m\}$. Also, given a set of classes $C = \{c_1, c_2, \dots, c_k\}$.

A decision tree T is a tree associated with D that has the following properties:

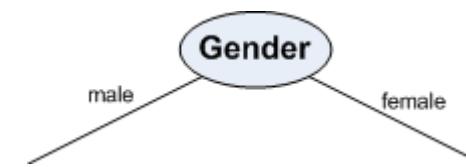
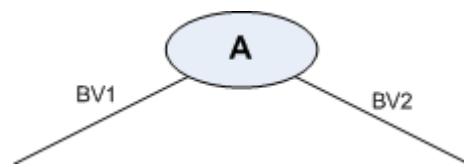
- Each internal node is labeled with an attribute A_i
- Each edges is labeled with predicate that can be applied to the attribute associated with the parent node of it
- Each leaf node is labeled with class c_j

Building Decision Tree

- In principle, there are exponentially many decision tree that can be constructed from a given database (also called training data).
 - Some of the tree may not be optimum
 - Some of them may give inaccurate result
- Two approaches are known
 - **Greedy strategy**
 - A top-down recursive divide-and-conquer
 - **Modification of greedy strategy**
 - ID3
 - C4.5
 - CART, etc.

Node Splitting in DT Algorithm

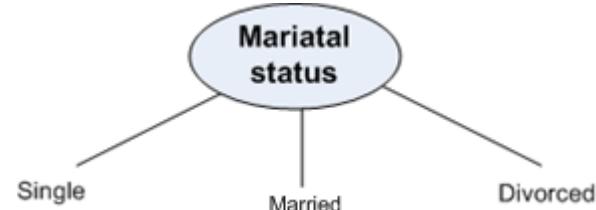
- DT algorithm must provides a method for expressing **an attribute test condition** and **corresponding outcome** for different attribute type
- **Case: Binary attribute**
 - This is the simplest case of node splitting
 - The test condition for a binary attribute generates only two outcomes



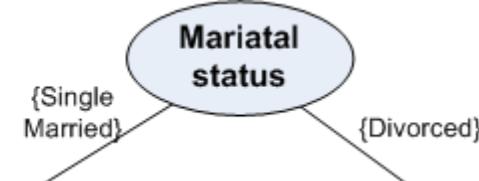
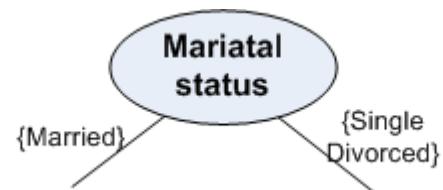
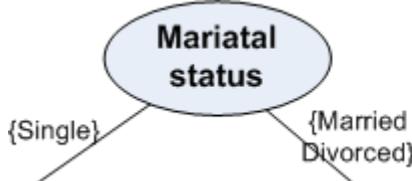
Node Splitting in DT Algorithm

- **Case: Nominal attribute**

- Since a nominal attribute can have many values, its test condition can be expressed in two ways:
 - A multi-way split
 - A binary split
- **Muti-way split:** Outcome depends on the number of distinct values for the corresponding attribute



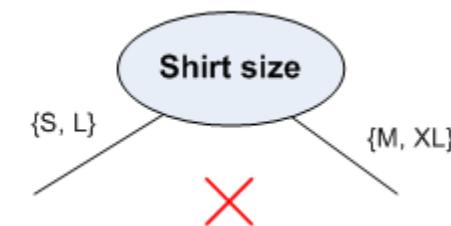
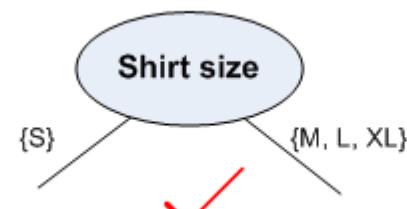
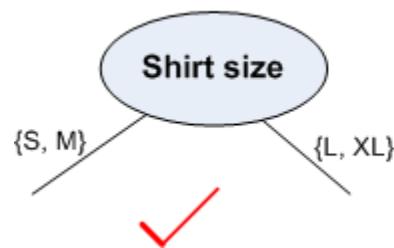
- **Binary splitting** by grouping attribute values



Node Splitting in DT Algorithm

- **Case: Ordinal attribute**

- It also can be expressed in two ways:
 - A multi-way split
 - A binary split
- **Multi-way split:** It is same as in the case of nominal attribute
- **Binary splitting** attribute values should be grouped maintaining the **order** property of the attribute values



Node Splitting in DT Algorithm

- **Case: Numerical attribute**

- For numeric attribute (with discrete or continuous values), a test condition can be expressed as a comparison set
 - **Binary outcome:** $A > v$ or $A \leq v$
 - In this case, decision tree induction must consider all possible split positions
 - **Range query :** $v_i \leq A < v_{i+1}$ for $i = 1, 2, \dots, q$ (if q number of ranges are chosen)
 - Here, q should be decided a priori
- For a numeric attribute, decision tree induction is a combinatorial optimization problem

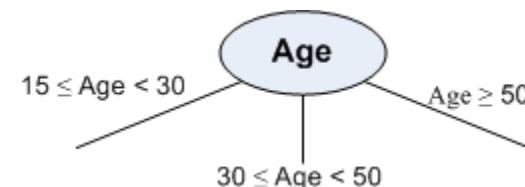


Illustration : DT Algorithm

Example 8.4: Illustration of BuildDT Algorithm

- Consider a training data set as shown.

Person	Gender	Height	Class
1	F	1.6	S
2	M	2.0	M
3	F	1.9	M
4	F	1.88	M
5	F	1.7	S
6	M	1.85	M
7	F	1.6	S
8	M	1.7	S
9	M	2.2	T
10	M	2.1	T
11	F	1.8	M
12	M	1.95	M
13	F	1.9	M
14	F	1.8	M
15	F	1.75	S

Attributes:

Gender = {Male(M), Female (F)} // Binary attribute

Height = {1.5, ..., 2.5} // Continuous attribute

Class = {Short (S), Medium (M), Tall (T)}

Given a person, we are to test in which class s/he belongs

Illustration : DT Algorithm

- To built a decision tree, we can select an attribute in two different orderings: $\langle \text{Gender}, \text{Height} \rangle$ or $\langle \text{Height}, \text{Gender} \rangle$
- Further, for each ordering, we can choose different ways of splitting
- Different instances are shown in the following.
- **Approach 1 : $\langle \text{Gender}, \text{Height} \rangle$**

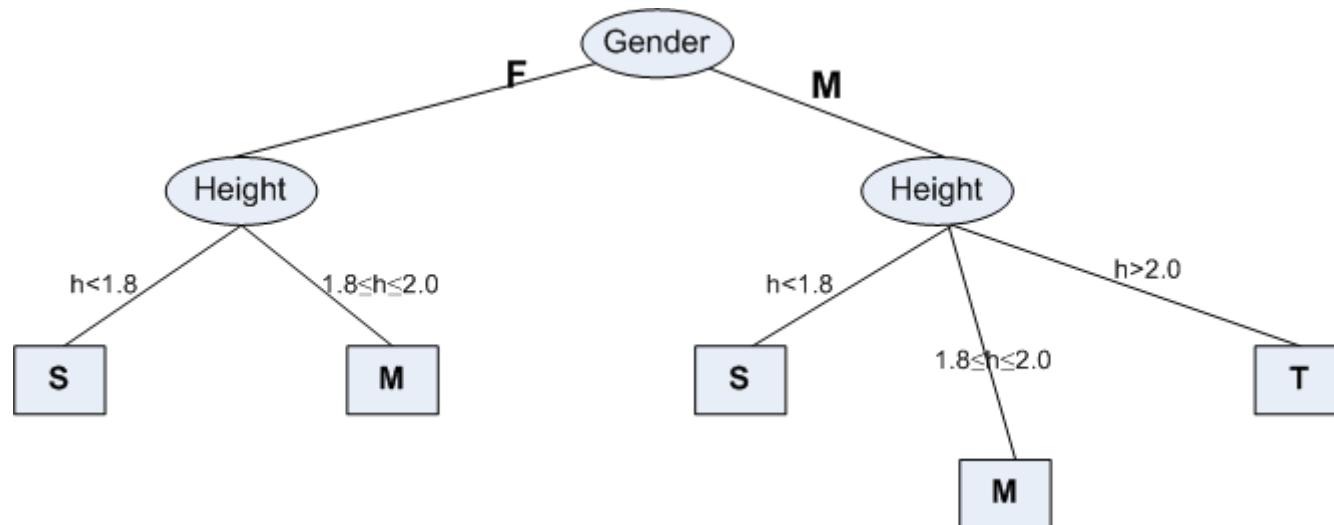


Illustration : DT Algorithm

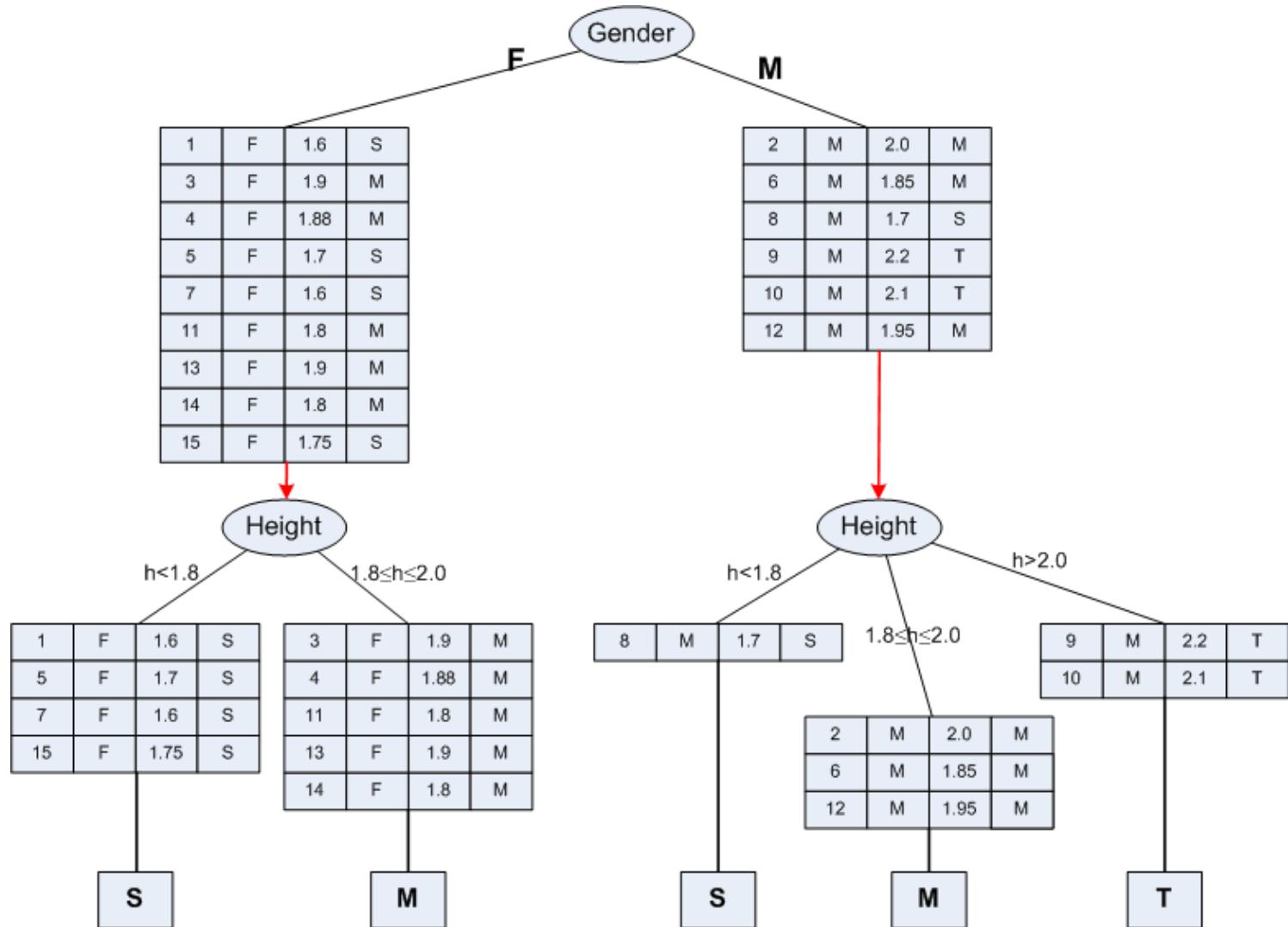


Illustration : DT Algorithm

- Approach 2 : <Height, Gender>

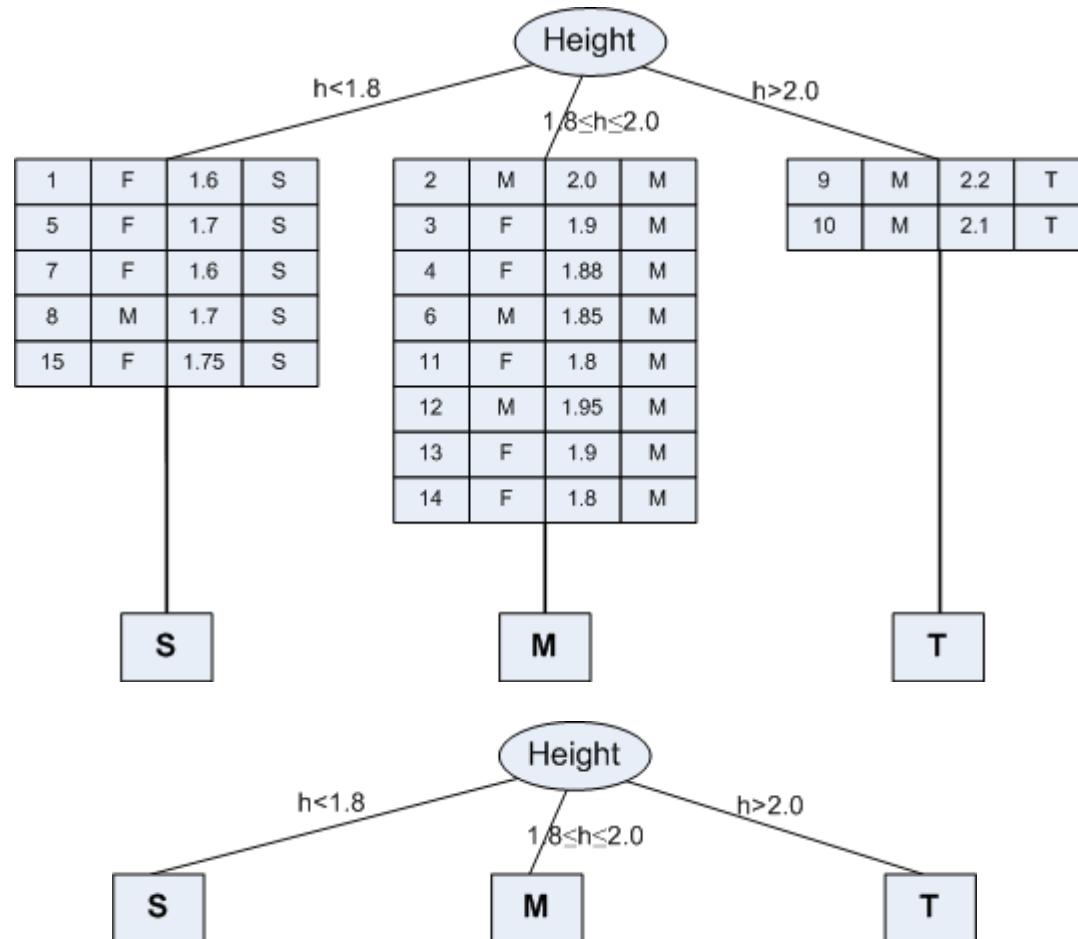


Illustration : DT Algorithm

Example 8.5: Illustration of DT Algorithm

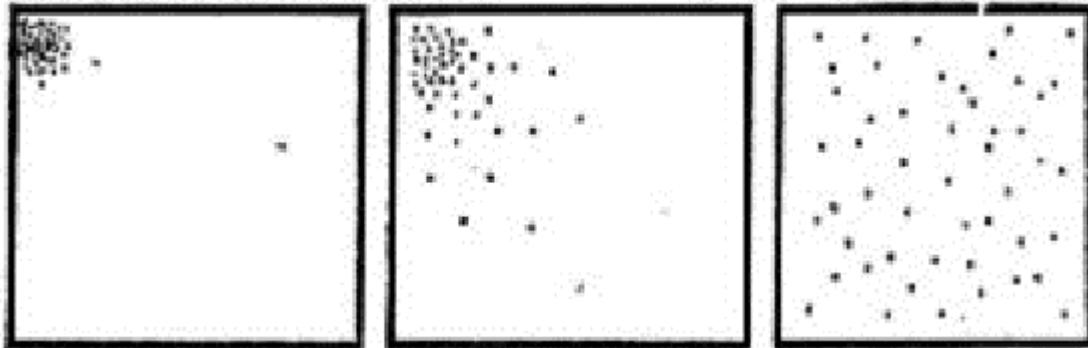
- Consider an anonymous database as shown.

A1	A2	A3	A4	Class
a11	a21	a31	a41	C1
a12	a21	a31	a42	C1
a11	a21	a31	a41	C1
a11	a22	a32	a41	C2
a11	a22	a32	a41	C2
a12	a22	a31	a41	C1
a11	a22	a32	a41	C2
a11	a22	a31	a42	C1
a11	a21	a32	a42	C2
a11	a22	a32	a41	C2
a12	a22	a31	a41	C1
a12	a22	a31	a42	C1

- Is there any “clue” that enables to select the “best” attribute first?
- Suppose, following are two attempts:
 - $A1 \rightarrow A2 \rightarrow A3 \rightarrow A4$ [naïve]
 - $A3 \rightarrow A2 \rightarrow A4 \rightarrow A1$ [Random]
- Draw the decision trees in the above-mentioned two cases.
- Are the trees different to classify any test data?
- If any other sample data is added into the database, is that likely to alter the decision tree already obtained?

Concept of Entropy

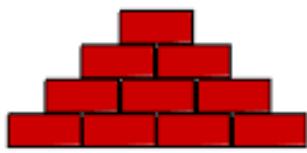
Concept of Entropy



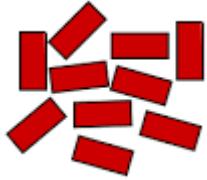
If a point represents a gas molecule,
then which system has the more
entropy?

How to measure?

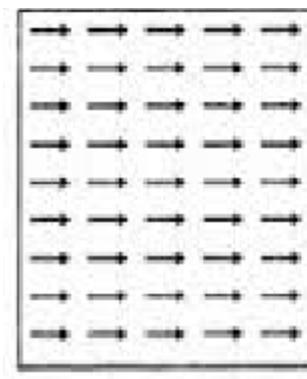
$$\Delta S = \frac{\Delta Q}{T} ?$$



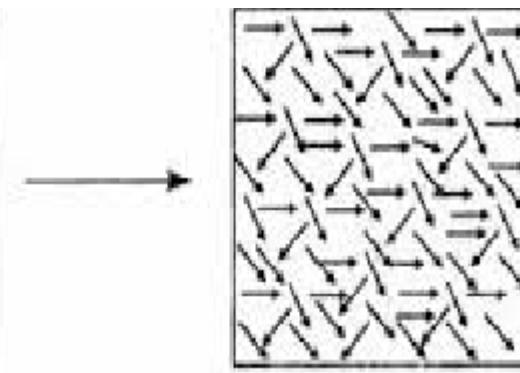
More ordered
less entropy



Less ordered
higher entropy



More organized or
ordered (less probable)



Less organized or
disordered (more probable)

Entropy and its Meaning

- Entropy is an important concept used in Physics in the context of heat and thereby uncertainty of the states of a matter.
- At a later stage, with the growth of Information Technology, entropy becomes an important concept in **Information Theory**.
- To deal with the classification job, entropy is an important concept, which is considered as
 - an information-theoretic measure of the “uncertainty” contained in a training data
 - due to the presence of more than one classes.

Entropy in Information Theory

- The entropy concept in information theory first time coined by Claude Shannon (1850).
- The first time it was used to measure the “information content” in messages.
- According to his concept of entropy, presently entropy is widely being used as a way of representing messages for efficient transmission by Telecommunication Systems.

Measure of Information Content

- People, in general, are information hungry!
- Everybody wants to acquire information (from newspaper, library, nature, fellows, etc.)
 - Think how a crime detector do it to know about the crime from crime spot and criminal(s).
 - Kids annoy their parents asking questions.
- In fact, fundamental thing is that we gather information asking questions (and decision tree induction is no exception).
 - We may note that information gathering may be with certainty or uncertainty.

Measure of Information Content

Example 8.6

- a) Guessing a birthday of your classmate

It is with uncertainty $\sim \frac{1}{365}$

Whereas guessing the day of his/her birthday is $\frac{1}{7}$.

This uncertainty, we may say varies between 0 to 1, both inclusive.

- b) As another example, a question related to event with eventuality (or impossibility) will be answered with 0 or 1 uncertainty.

- Does sun rises in the East? (answer is with 0 uncertainty)
- Will mother give birth to male baby? (answer is with $\frac{1}{2}$ uncertainty)
- Is there a planet like earth in the galaxy? (answer is with an extreme uncertainty)

Entropy Calculation

- If there are m objects with frequencies p_1, p_2, \dots, p_m , then the average number of bits (i.e. questions) that need to be examined a value, that is, entropy is the frequency of occurrence of the i^{th} value multiplied by the number of bits that need to be determined, summed up values of i from 1 to m .

Theorem: Entropy calculation

If p_i denotes the frequencies of occurrences of m distinct objects, then the entropy E is

$$E = \sum_{i=1}^m p_i \log(1/p_i) \text{ and } \sum_{i=1}^m p_i = 1$$

Note:

- If all are equally likely, then $p_i = \frac{1}{m}$ and $E = \log_2 m$; it is the special case.

Entropy of a Training Set

- If there are k classes c_1, c_2, \dots, c_k and p_i for $i = 1 \text{ to } k$ denotes the number of occurrences of classes c_i divided by the total number of instances (i.e., the frequency of occurrence of c_i) in the training set, then entropy of the training set is denoted by

$$E = - \sum_{i=1}^m p_i \log_2 p_i$$

Here, E is measured in “bits” of information.

Note:

- The above formula should be summed over the non-empty classes only, that is, classes for which $p_i \neq 0$
- E is always a positive quantity
- E takes its minimum value (zero) if and only if all the instances have the same class (i.e., the training set with only one non-empty class, for which the probability 1).
- Entropy takes its maximum value when the instances are equally distributed among k possible classes. In this case, the maximum value of E is $\log_2 k$.

Entropy of a Training Set

Example 18.10: OPTH dataset

Consider the OTPH data shown in the following table with total 24 instances in it.

Age	Eye sight	Astigmatic	Use Type	Class
1	1	1	1	3
	1	1	2	2
	1	2	1	3
	1	2	2	1
	2	1	1	3
	2	1	2	2
2	2	2	1	3
	2	2	2	1
	2	1	1	3
	2	1	2	2
	2	1	2	3
	1	2	1	1
3	2	2	1	3
	2	2	2	2
	2	2	1	3
	3	1	1	3
	3	1	2	3
	3	2	1	3
3	3	2	2	1
	3	3	2	2
	3	2	1	3
	3	2	2	2
	3	2	1	3
	3	2	2	3

A coded forms for all values of attributes are used to avoid the cluttering in the table.

Entropy of a training set

Specification of the attributes are as follows.

Age	Eye Sight	Astigmatic	Use Type
1: Young	1: Myopia	1: No	1: Frequent
2: Middle-aged	2: Hypermetropia	2: Yes	2: Less
3: Old			

Class: **1: Contact Lens** **2:Normal glass** **3: Nothing**

In the OPTH database, there are 3 classes and 4 instances with class 1, 5 instances with class 2 and 15 instances with class 3. Hence, entropy E of the database is:

$$E = -\frac{4}{24} \log_2 \frac{4}{24} - \frac{5}{24} \log_2 \frac{5}{24} - \frac{15}{24} \log_2 \frac{15}{24} = 1.3261$$

Note:

- The entropy of a training set implies the number of yes/no questions, on the average, needed to determine an unknown test to be classified.
- It is very crucial to decide the series of questions about the value of a set of attribute, which collectively determine the classification. Sometimes it may take one question, sometimes many more.
- Decision tree induction helps us to ask such a series of questions. In other words, we can utilize entropy concept to build a better decision tree.

How entropy can be used to build a decision tree ?

Decision Tree Induction Techniques

- Decision tree induction is a top-down, recursive and divide-and-conquer approach.
- The procedure is to choose an attribute and split it into from a larger training set into smaller training sets.
- Different algorithms have been proposed to take a good control over
 1. Choosing the best attribute to be splitted, and
 2. Splitting criteria
- Several algorithms have been proposed for the above tasks. In this lecture, we shall limit our discussions into three important of them
 - **ID3**
 - **C 4.5**
 - **CART**

Algorithm ID3

ID3: Decision Tree Induction Algorithms

- Quinlan [1984] introduced the ID3, a popular short form of Iterative Dichotomizer 3 for decision trees from a set of training data.
- In ID3, each node corresponds to a splitting attribute and each arc is a possible value of that attribute.
- At each node, the splitting attribute is selected to be the most informative among the attributes not yet considered in the path starting from the root.

Algorithm ID3

- In ID3, **entropy is used** to measure how informative a node is.
 - It is observed that splitting on any attribute has **the property that average entropy of the resulting training subsets will be less than or equal to that of the previous training set.**
 - ID3 algorithm defines a measurement of a splitting called **Information Gain** to determine the goodness of a split.
 - The attribute with the **largest value of information gain** is chosen as the splitting attribute and
 - it partitions into a number of smaller training sets based on the **distinct values of attribute** under split.

Defining Information Gain

- We consider the following symbols and terminologies to define information gain, which is denoted as α .
- $D \equiv$ denotes the training set at any instant
- $|D| \equiv$ denotes the size of the training set D
- $E(D) \equiv$ denotes the entropy of the training set D
- The entropy of the training set D

$$E(D) = -\sum_{i=1}^k p_i \log_2(p_i)$$

- where the training set D has c_1, c_2, \dots, c_k , the k number of distinct classes and
- $p_i, 0 < p_i \leq 1$ is the probability that an arbitrary tuple in D belongs to class c_i ($i = 1, 2, \dots, k$).

Defining Information Gain

Definition 2: Weighted Entropy

The weighted entropy denoted as $E_A(D)$ for all partitions of D with respect to A is given by:

$$E_A(D) = \sum_{j=1}^m \frac{|D_j|}{|D|} E(D_j)$$

Here, the term $\frac{|D_j|}{|D|}$ denotes the weight of the j -th training set.

More meaningfully, $E_A(D)$ is the expected information required to classify a tuple from D based on the splitting of A .

Defining Information Gain

- Our objective is to take A on splitting to produce an exact classification (also called pure), that is, all tuples belong to one class.
- However, it is quite likely that the partitions is impure, that is, they contain tuples from two or more classes.
- In that sense, $E_A(D)$ is a measure of impurities (or purity). A lesser value of $E_A(D)$ implying more power the partitions are.

Definition 3: Information Gain

Information gain, $\alpha(A, D)$ of the training set D splitting on the attribute A is given by

$$\alpha(A, D) = E(D) - E_A(D)$$

In other words, $\alpha(A, D)$ gives us an estimation how much would be gained by splitting on A . The attribute A with the highest value of α should be chosen as the splitting attribute for D .

Information Gain Calculation

Example 8.11 : Information gain on splitting OPTH

- Let us refer to the OPTH database discussed earlier.
- Splitting on **Age** at the root level, it would give three subsets D_1, D_2 and D_3 as shown in the tables in the following three slides.
- The entropy $E(D_1), E(D_2)$ and $E(D_3)$ of training sets D_1, D_2 and D_3 and corresponding weighted entropy $E_{Age}(D_1), E_{Age}(D_2)$ and $E_{Age}(D_3)$ are also shown alongside.
- The Information gain $\alpha(Age, OPTH)$ is then can be calculated as **0.0394**.
- Recall that entropy of OPTH data set, we have calculated as $E(OPTH) = \textcolor{red}{1.3261}$
(see Slide #29)

Information Gain Calculation

Example 8.11 : Information gain on splitting OPTH

Training set: $D_1(\text{Age} = 1)$

Age	Eye-sight	Astigmatism	Use type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1

$$E(D_1) = -\frac{2}{8} \log_2\left(\frac{2}{8}\right) - \frac{2}{8} \log_2\left(\frac{2}{8}\right) - \frac{4}{8} \log_2\left(\frac{4}{8}\right) = 1.5$$

$$E_{Age}(D_1) = \frac{8}{24} \times 1.5 = 0.5000$$

Calculating Information Gain

Training set: $D_2(\text{Age} = 2)$

Age	Eye-sight	Astigmatism	Use type	Class
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3

$$E(D_2) = -\frac{1}{8} \log_2\left(\frac{1}{8}\right) - \frac{2}{8} \log_2\left(\frac{2}{8}\right) - \frac{5}{8} \log_2\left(\frac{5}{8}\right)$$
$$= 1.2988$$

$$E_{\text{Age}}(D_2) = \frac{8}{24} \times 1.2988 = 0.4329$$

Calculating Information Gain

Training set: $D_3(\text{Age} = 3)$

Age	Eye-sight	Astigmatism	Use type	Class
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	1
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

$$E(D_3) = -\frac{1}{8} \log_2\left(\frac{1}{8}\right) - \frac{1}{8} \log_2\left(\frac{1}{8}\right) - \frac{6}{8} \log_2\left(\frac{6}{8}\right) = 1.0613$$

$$E_{\text{Age}}(D_3) = \frac{8}{24} \times 1.0613 = 0.3504$$

$$\alpha(\text{Age}, D) = 1.3261 - (0.5000 + 0.4329 + 0.3504) = \mathbf{0.0394}$$

Information Gains for Different Attributes

- In the same way, we can calculate the information gains, when splitting the OPTH database on **Eye-sight**, **Astigmatic** and **Use Type**. The results are summarized below.
- Splitting attribute: **Age**

$$\alpha(Age, OPTH) = 0.0394$$

- Splitting attribute: **Eye-sight**

$$\alpha(Eye_sight, OPTH) = 0.0395$$

- Splitting attribute: **Astigmatic**

$$\alpha(Astigmatic, OPTH) = 0.3770$$

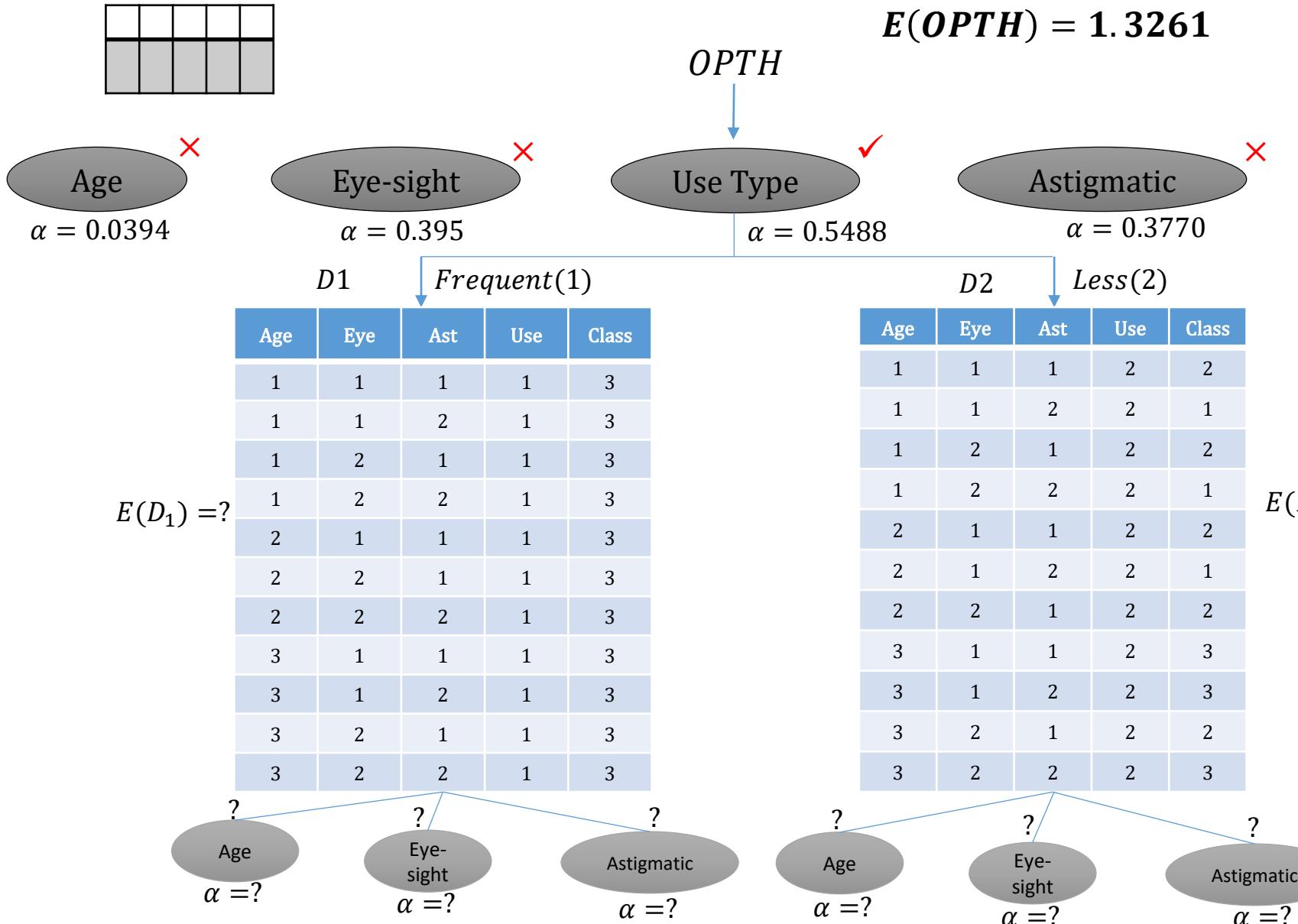
- Splitting attribute: **Use Type**

$$\alpha(Use\ Type, OPTH) = 0.5488$$

Decision Tree Induction : ID3 Way

- The ID3 strategy of attribute selection is to choose to split on the attribute that gives the greatest reduction in the weighted average entropy
 - The one that maximizes the value of information gain
- In the example with OPTH database, the larger values of information gain is $\alpha(\text{Use Type}, OPTH) = 0.5488$
 - Hence, the attribute should be chosen for splitting is “[Use Type](#)”.
- The process of splitting on nodes is repeated for each branch of the evolving decision tree, and the final tree, which would look like is shown in the following slide and calculation is left for practice.

Decision Tree Induction : ID3 Way



Frequency Table : Calculating α

- Calculation of entropy for each table and hence information gain for a particular split appears tedious (at least manually)!
- As an alternative, we discuss **a short-cut method** of doing the same using a special data structure called **Frequency Table**.
- **Frequency Table:** Suppose, $X = \{x_1, x_2, \dots, x_n\}$ denotes an attribute with $n - \text{different}$ attribute values in it. For a given database D , there are a set of k classes say $C = \{c_1, c_2, \dots, c_k\}$. Given this, a frequency table will look like as follows.

Frequency Table : Calculating α

	X					
	x_1	x_2	x_i	x_n
c_1			
c_2			
\vdots	\vdots	\vdots	\vdots	\vdots
c_j			f_{ij}	
\vdots	\vdots	\vdots	\vdots	\vdots
c_k			

- Number of rows = Number of classes
- Number of columns = Number of attribute values
- f_{ij} = Frequency of x_i for class c_j

Assume that $|D| = N$, the number of total instances of D .

Calculation of α using Frequency Table

Example 8.12 : OTPH Dataset

With reference to OPTH dataset, and for the attribute Age, the frequency table would look like

	Age=1	Age=2	Age=3	Row Sum
Class 1	2	1	1	4
Class 2	2	2	1	5
Class 3	4	5	6	15
Column Sum	8	8	8	24

Column Sums

N=24

Calculation of α using Frequency Table

- The weighted average entropy $E_X(D)$ then can be calculated from the frequency table following the
 - Calculate $V = f_{ij} \log_2 f_{ij}$ for all $i = 1, 2, \dots, k$
(Entry Sum) $j = 1, 2, \dots, n$ and $v_{ij} \neq 0$
 - Calculate $S = s_i \log_2 s_i$ for all $i = 1, 2, \dots, n$
(Column Sum) in the row of column sum
 - Calculate $E_X(D) = (-V + S)/N$

Example 8.13: OTPH Dataset

For the frequency table in Example 18.12, we have

$$V$$

$$= 2 \log 2 + 1 \log 1 + 1 \log 1 + 2 \log 2 + 2 \log 2 + 1 \log 1 + 4 \log 4 + 5 \log 5 + 6 \log 6$$
$$S = 8 \log 8 + 8 \log 8 + 8 \log 8$$

$$E_{Age}(OPTH) = 1.2867$$

LIMITING VALUES OF INFORMATION GAIN

- The Information gain metric used in ID3 always should be positive or zero.
- It is always positive value because information is always gained (i.e., purity is improved) by splitting on an attribute.
- On the other hand, when a training set is such that if there are k classes, and the entropy of training set takes the largest value i.e., $\log_2 k$ (this occurs when the classes are balanced), then the information gain will be zero.

Limiting Values of Information Gain

Example 8.14: Limiting values of Information gain

Consider a training set shown below.

Data set <i>Table A</i>		
X	Y	Class
1	1	A
1	2	B
2	1	A
2	2	B
3	2	A
3	1	B
4	2	A
4	1	B

		X		Table X
	1	2	3	4
A	1	1	1	1
B	1	1	1	1
C.Sum	2	2	2	2

Frequency table of X

	Y		Table Y
	1	2	
A	2	2	
B	2	2	
C.Sum	4	4	

Frequency table of Y

Limits of Information Gain

- Entropy of Table A is
$$E = -\frac{1}{2} \log \frac{1}{2} - \frac{1}{2} \log \frac{1}{2} = \log 2 = 1 \text{ (The maximum entropy).}$$
- In this example, whichever attribute is chosen for splitting, each of the branches will also be balanced thus each with maximum entropy.
- In other words, information gain in both cases (i.e., splitting on X as well as Y) will be zero.

Note:

- The absence of information gain does not imply that there is no profit for splitting on the attribute.
- Even if it is chosen for splitting, ultimately it will lead to a final decision tree with the branches terminated by a leaf node and thus having an entropy of zero.
- Information gain can never be a negative value.

Reference

- The detail material related to this lecture can be found in

Data Mining: Concepts and Techniques, (3rd Edn.), Jiawei Han, Micheline Kamber, Morgan Kaufmann, 2015.

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, Addison-Wesley, 2014

Any question?



Brain Computer Interaction

Decision Tree Induction – CART & C4.5

Course Instructors

Dr. Sreeja S R

Assistant Professor

**Indian Institute of Information Technology
IIIT Sri City**

Algorithm CART

CART Algorithm

- It is observed that information gain measure used in ID3 **is biased towards test with many outcomes**, that is, it prefers to select attributes having a large number of values.
- L. Breiman, J. Friedman, R. Olshen and C. Stone in 1984 proposed an algorithm to build a binary decision tree also called CART decision tree.
 - CART stands for **Classification and Regression Tree**
 - In fact, invented independently at the same time as ID3 (1984).
 - ID3 and CART are two cornerstone algorithms spawned a flurry of work on decision tree induction.
- CART is a technique that generates **a binary decision tree**; That is, unlike ID3, in CART, for each node only two children are created.
- ID3 uses Information gain as a measure to select the best attribute to be splitted, whereas CART do the same but using another measurement called **Gini index**. It is also known as **Gini Index of Diversity** and is denoted as γ .

Gini Index of Diversity

Definition 20.1: Gini Index

Suppose, D is a training set with size $|D|$ and $C = \{c_1, c_2, \dots, c_k\}$ be the set of k classifications and $A = \{a_1, a_2, \dots, a_m\}$ be any attribute with m different values of it. Like entropy measure in ID3, CART proposes Gini Index (denoted by G) as the measure of impurity of D . It can be defined as follows.

$$G(D) = 1 - \sum_{i=1}^k p_i^2$$

where p_i is the probability that a tuple in D belongs to class c_i and p_i can be estimated as

$$p_i = \frac{|C_{i,D}|}{D}$$

where $|C_{i,D}|$ denotes the number of tuples in D with class c_i .

Gini Index of Diversity

Note

- $G(D)$ measures the “impurity” of data set D .
- The smallest value of $G(D)$ is zero
 - which it takes when all the classifications are same.
- It takes its largest value = $1 - \frac{1}{k}$
 - when the classes are evenly distributed between the tuples, that is the frequency of each class is $\frac{1}{k}$.

Gini Index of Diversity

Definition 20.2: Gini Index of Diversity

Suppose, a binary partition on A splits D into D_1 and D_2 , then the **weighted average Gini Index of splitting** denoted by $G_A(D)$ is given by

$$G_A(D) = \frac{|D_1|}{D} \cdot G(D_1) + \frac{|D_2|}{D} \cdot G(D_2)$$

This binary partition of D reduces the impurity and the reduction in impurity is measured by

$$\gamma(A, D) = G(D) - G_A(D)$$

Gini Index of Diversity and CART

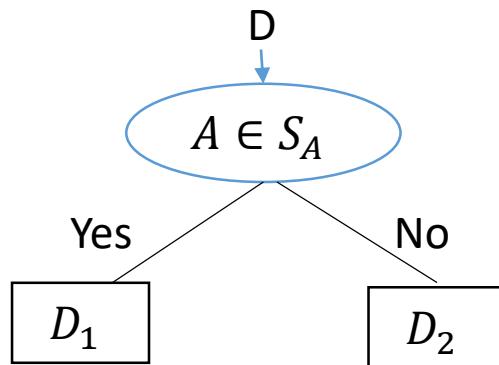
- This $\gamma(A, D)$ is called the Gini Index of diversity.
- It is also called as “impurity reduction”.
- The attribute that **maximizes** the reduction in impurity (or equivalently, has the **minimum value of $G_A(D)$**) is selected for the attribute to be splitted.

n-ary Attribute Values to Binary Splitting

- The CART algorithm considers a binary split for each attribute.
- We shall discuss how the same is possible for attribute with more than two values.
- **Case 1: Discrete valued attributes**
- Let us consider the case where A is a discrete-valued attribute having m discrete values a_1, a_2, \dots, a_m .
- To determine the best binary split on A , we examine all of the possible subsets say 2^A of A that can be formed using the values of A .
- Each subset $S_A \in 2^A$ can be considered as a binary test for attribute A of the form " $A \in S_A?$ ".

n-ary Attribute Values to Binary Splitting

- Thus, given a data set D , we have to perform a test for an attribute value A like

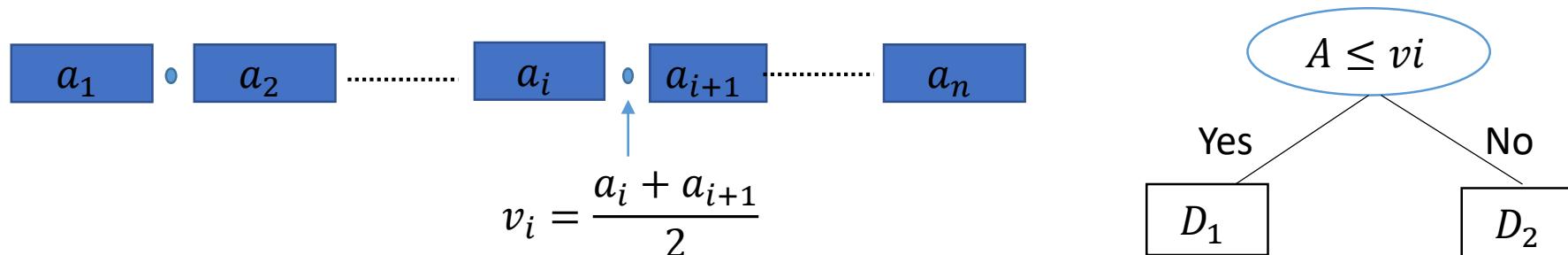


- This test is satisfied if the value of A for the tuples is among the values listed in S_A .
- If A has m distinct values in D , then there are 2^m possible subsets, out of which the empty subset $\{\}$ and the power set $\{a_1, a_2, \dots, a_n\}$ should be excluded (as they really do not represent a split).
- Thus, there are $2^m - 2$ possible ways to form two partitions of the dataset D , based on the binary split of A .

n-ary Attribute Values to Binary Splitting

Case2: Continuous valued attributes

- For a continuous-valued attribute, each possible split point must be taken into account.
- According to that strategy, the mid-point between a_i and a_{i+1} , let it be v_i , then



n-ary Attribute Values to Binary Splitting

- Each pair of (sorted) adjacent values is taken as a possible split-point say v_i .
- D_1 is the set of tuples in D satisfying $A \leq v_i$ and D_2 in the set of tuples in D satisfying $A > v_i$.
- The point giving the minimum Gini Index $G_A(D)$ is taken as the split-point of the attribute A .

Note

- The attribute A and either its splitting subset S_A (for discrete-valued splitting attribute) or split-point v_i (for continuous valued splitting attribute) together form the splitting criteria.

CART Algorithm : Illustration

Example 20.1 : CART Algorithm

Suppose we want to build decision tree for the data set EMP as given in the table below.

	Tuple#	Age	Salary	Job	Performance	Select
Age Y : young M : middle-aged O : old	1	Y	H	P	A	N
	2	Y	H	P	E	N
	3	M	H	P	A	Y
Salary L : low M : medium H : high	4	O	M	P	A	Y
	5	O	L	G	A	Y
	6	O	L	G	E	N
	7	M	L	G	E	Y
Job G : government P : private	8	Y	M	P	A	N
	9	Y	L	G	A	Y
Performance A : Average E : Excellent	10	O	M	G	A	Y
	11	Y	M	G	E	Y
	12	M	M	P	E	Y
Class : Select Y : yes N : no	13	M	H	G	A	Y
	14	O	M	P	E	N

CART Algorithm : Illustration

For the EMP data set,

$$\begin{aligned} G(EMP) &= 1 - \sum_{i=1}^2 p_i^2 \\ &= 1 - \left[\left(\frac{9}{14} \right)^2 + \left(\frac{5}{14} \right)^2 \right] \\ &= \mathbf{0.4592} \end{aligned}$$

Now let us consider the calculation of $G_A(EMP)$ for **Age**, **Salary**, **Job** and **Performance**.

CART Algorithm : Illustration

Attribute of splitting: Age

The attribute age has three values, namely Y, M and O. So there are 6 subsets, that should be considered for splitting as:

$$G_{age_1}(D) = \frac{5}{14} * \left(1 - \left(\frac{3}{5} \right)^2 - \left(\frac{2}{5} \right)^2 \right) + \frac{9}{14} \left(1 - \left(\frac{6}{14} \right)^2 - \left(\frac{8}{14} \right)^2 \right) = 0.4862$$

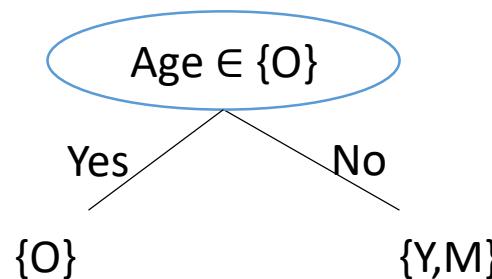
$$G_{age_2}(D) = ?$$

$$G_{age_3}(D) = ?$$

$$G_{age_4}(D) = G_{age_3}(D)$$

$$G_{age_5}(D) = G_{age_2}(D)$$

$$G_{age_6}(D) = G_{age_1}(D)$$



The best value of Gini Index while splitting attribute Age is $\gamma(Age_3, D) = 0.3750$

CART Algorithm : Illustration

Attribute of Splitting: Salary

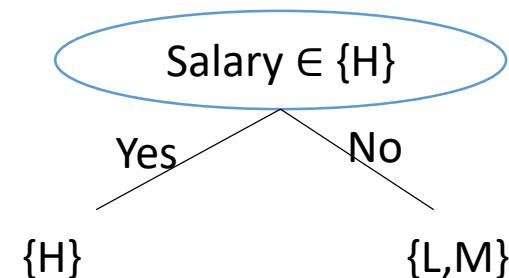
The attribute salary has three values namely L , M and H . So, there are 6 subsets, that should be considered for splitting as:

$$\begin{array}{llllll} \{L\} & \{M, H\} & \{M\} & \{L, H\} & \{H\} & \{L, M\} \\ sal_1' & sal_2' & sal_3' & sal_4' & sal_5' & sal_6 \end{array}$$

$$G_{sal_1}(D) = G_{sal_2}(D) = 0.3000$$

$$G_{sal_3}(D) = G_{sal_4}(D) = 0.3150$$

$$G_{sal_5}(D) = G_{sal_6}(D) = 0.4508$$



$$\gamma(salary_{(5,6)}, D) = 0.4592 - 0.4508 = 0.0084$$

CART Algorithm : Illustration

Attribute of Splitting: job

Job being the binary attribute , we have

$$G_{job}(D) = \frac{7}{14} G(D_1) + \frac{7}{14} G(D_2)$$

$$= \frac{7}{14} \left[1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2 \right] + \frac{7}{14} \left[1 - \left(\frac{6}{7}\right)^2 - \left(\frac{1}{7}\right)^2 \right] = ?$$

$$\gamma(job, D) = ?$$

CART Algorithm : Illustration

Attribute of Splitting: Performance

Job being the binary attribute , we have

$$G_{Performance}(D) = ?$$

$$\gamma(performance, D) = ?$$

Out of these $\gamma(salary, D)$ gives the minimum value and hence, the attribute **Salary** would be chosen for splitting subset $\{M, H\}$ or $\{L\}$.

Note:

It can be noted that the procedure following “information gain” calculation (i.e. $\propto (A, D)$) and that of “impurity reduction” calculation (i.e. $\gamma(A, D)$) are near about.

Calculating γ using Frequency Table

- We have learnt that splitting on an attribute gives a reduction in the average Gini Index of the resulting subsets (as it does for entropy).
- Thus, in the same way the average weighted Gini Index can be calculated using the same frequency table used to calculate information gain $\alpha(A, D)$, which is as follows.

The $G(D_j)$ for the j^{th} subset D_j

$$G(D_j) = 1 - \sum_{i=1}^k \left(\frac{f_{ij}}{|D_j|} \right)^2$$

Calculating γ using Frequency Table

The average weighted Gini Index, $G_A(D_j)$ (assuming that attribute has m distinct values is)

$$\begin{aligned}G_A(D_j) &= \sum_{j=1}^k \frac{|D_j|}{|D_1|} \cdot G(D_j) \\&= \sum_{j=1}^m \frac{|D_j|}{|D|} - \sum_{j=1}^m \sum_{i=1}^k \frac{|D_j|}{|D|} \cdot \left(\frac{f_{ij}}{|D_j|} \right)^2 \\&= 1 - \frac{1}{|D|} \sum_{j=1}^m \frac{1}{D_j} \cdot \sum_{i=1}^k f_{ij}^2\end{aligned}$$

The above gives a formula for m -attribute values; however, it can be fine tuned to subset of attributes also.

Illustration: Calculating γ using Frequency Table

Example 20.2 : Calculating γ using frequency table of OPTH

Let us consider the frequency table for OPTH database considered earlier. Also consider the attribute A_1 with three values 1, 2 and 3. The frequency table is shown below.

	1	2	3
Class 1	2	1	1
Class 2	2	2	1
Class 3	4	5	6
Column sum	8	8	8

Illustration: Calculating γ using Frequency Table

Now we can calculate the value of Gini Index with the following steps:

1. For each non-empty column, form the sum of the squares of the values in the body of the table and divide by the column sum.
2. Add the values obtained for all columns and divided by $|D|$, the size of the database.
3. Subtract the total from 1.

As an example, with reference to the frequency table as mentioned just.

$$A_1 = 1 = \frac{(2^2 + 2^2 + 4^2)}{24} = 3.0$$

$$A_1 = 2 = \frac{(1^2 + 2^2 + 5^2)}{24} = 3.75$$

$$A_1 = 3 = \frac{(1^2 + 1^2 + 6^2)}{24} = 4.75$$

$$\text{So, } G_{A1}(D) = 1 - \frac{3+3.75+4.75}{24} = 0.5208$$

Illustration: Calculating γ using Frequency Table

Thus, the reduction in the value of Gini Index on splitting attribute A_1 is

$$\gamma(A_1, D) = 0.5382 - 0.5208 = 0.0174$$

where $G(D) = 0.5382$

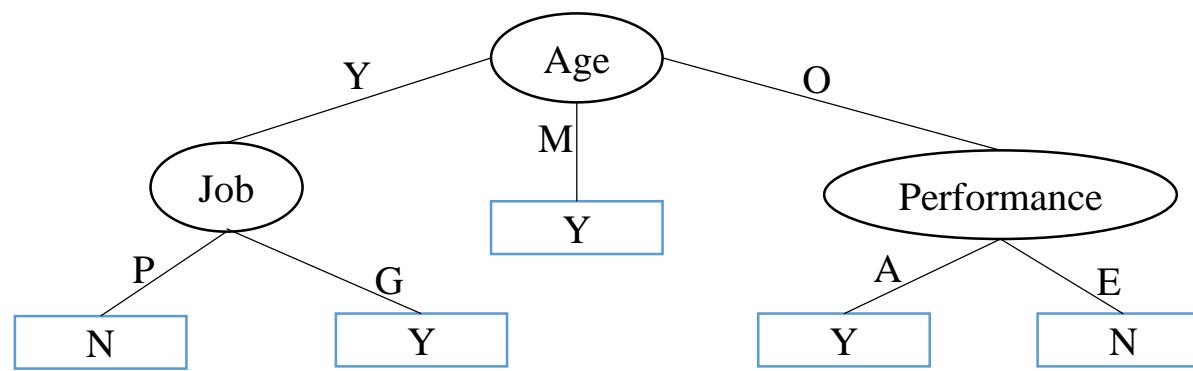
The calculation can be extended to other attributes in the OTPH database and is left as an exercise.



Decision Trees with ID3 and CART Algorithms

Example 20.3 : Comparing Decision Trees of EMP Data set

Compare two decision trees obtained using ID3 and CART for the EMP dataset. The decision tree according to ID3 is given for your ready reference (subject to the verification)



Decision Tree using ID3

?

Decision Tree using CART

Algorithm C4.5

Algorithm C4.5 : Introduction

- J. Ross Quinlan, a researcher in machine learning developed a decision tree induction algorithm in 1984 known as ID3 (Iterative Dichotomizer 3).
- Quinlan later presented C4.5, a successor of ID3, addressing some limitations in ID3.
- ID3 uses information gain measure, which is, in fact **biased towards splitting attribute having a large number of outcomes**.
- For example, if an attribute has distinct values for all tuples, then it would result in a large number of partitions, each one containing just one tuple.
 - In such a case, note that each partition is pure, and hence the purity measure of the partition, that is $E_A(D) = 0$

Algorithm C4.5 : Introduction

Example 20.4 : Limitation of ID3

In the following, each tuple belongs to a unique class. The splitting on A is shown.

A	-----	class
a_1		
a_2		
⋮		
a_j		
⋮		
a_n		

$E(D_j) = l \log_2 l = 0$

a_1	-----	
a_2	-----	
⋮		
a_j		
a_n	-----	

$$E_A(D) = \sum_{j=1}^n \frac{|D_j|}{|D|} \cdot E(D_j) = \sum_{j=1}^n \frac{1}{|D|} \cdot 0 = 0$$

Thus, $\alpha(A, D) = E(D) - E_A(D)$ is maximum in such a situation.

Algorithm: C 4.5 : Introduction

- Although, the previous situation is an extreme case, intuitively, we can infer that **ID3 favours splitting attributes having a large number of values**
 - compared to other attributes, which have a less variations in their values.
- Such a partition appears to be useless for classification.
- This type of problem is called **overfitting problem**.

Note:

Decision Tree Induction Algorithm ID3 may suffer from overfitting problem.

Algorithm: C 4.5 : Introduction

- The overfitting problem in ID3 is due to the measurement of information gain.
- In order to reduce the effect of the use of the bias due to the use of information gain, C4.5 uses a different measure called **Gain Ratio**, denoted as β .
- Gain Ratio is a kind of normalization to information gain using a **split information**.

Algorithm: C4.5 : Gain Ratio

Definition 20.3: Gain Ratio

The gain ratio can be defined as follows. We first define **split information** $E_A^*(D)$ as

$$E_A^*(D) = - \sum_{j=1}^m \frac{|D_j|}{|D|} \cdot \log \frac{|D_j|}{|D|}$$

Here, m is the number of distinct values in A .

The gain ratio is then defined as $\beta(A, D) = \frac{\alpha(A, D)}{E_A^*(D)}$, where $\alpha(A, D)$ denotes the information gain on splitting the attribute A in the dataset D .

Physical Interpretation of $E_A^*(D)$

Split information $E_A^*(D)$

- The value of split information depends on
 - the number of (distinct) values an attribute has and
 - how uniformly those values are distributed.
- In other words, it represents the **potential information** generated by splitting a data set D into m partitions, corresponding to the m outcomes of on attribute A .
- Note that for each outcome, it considers the number of tuples having that outcome with respect to the total number of tuples in D .

Physical Interpretation of $E_A^*(D)$

Example 20.5 : Split information $E_A^*(D)$

- To illustrate $E_A^*(D)$, let us examine the case where there are 32 instances and splitting an attribute A which has a_1, a_2, a_3 and a_4 sets of distinct values.
 - Distribution 1 : Highly non-uniform distribution of attribute values

	a_1	a_2	a_3	a_4
Frequency	32	0	0	0

$$E_A^*(D) = - \frac{32}{32} \log_2 \left(\frac{32}{32} \right) = -\log_2 1 = 0$$

- Distribution 2

	a_1	a_2	a_3	a_4
Frequency	16	16	0	0

$$E_A^*(D) = - \frac{16}{32} \log_2 \left(\frac{16}{32} \right) - \frac{16}{32} \log_2 \left(\frac{16}{32} \right) = \log_2 2 = 1$$

Physical Interpretation of $E_A^*(D)$

Distribution 3

	a_1	a_2	a_3	a_4
Frequency	16	8	8	0

$$E_A^*(D) = -\frac{16}{32} \log_2\left(\frac{16}{32}\right) - \frac{8}{32} \log_2\left(\frac{8}{32}\right) - \frac{8}{32} \log_2\left(\frac{8}{32}\right) = 1.5$$

• Distribution 4

	a_1	a_2	a_3	a_4
Frequency	16	8	4	4

$$E_A^*(D) = 1.75$$

• Distribution 5: Uniform distribution of attribute values

	a_1	a_2	a_3	a_4
Frequency	8	8	8	8

$$E_A^*(D) = \left(-\frac{8}{32} \log_2\left(\frac{8}{32}\right)\right) * 4 = -\log_2\left(\frac{1}{4}\right) = 2.0$$

Physical Interpretation of $E_A^*(D)$

- In general, if there are m attribute values, each occurring equally frequently, then the split information is $\log_2 m$.
- Based on the Example 20.5, we can summarize our observation on split information as under:
 - Split information is 0 when there is a single attribute value. It is a trivial case and implies *the minimum possible value of split information*.
 - For a given data set, when instances are uniformly distributed with respect to the attribute values, split information increases as the number of different attribute values increases.
 - The maximum value of split information occur when there are many possible attribute values, all are equally frequent.

Note:

- Split information varies between 0 and $\log_2 m$ (both inclusive)

Physical Interpretation of $\beta(A, B)$

- Information gain signifies how much information will be gained on partitioning the values of attribute A
 - Higher information gain means splitting of A is more desirable.
- On the other hand, split information forms the denominator in the gain ratio formula.
 - This implies that higher the value of split information is, lower the gain ratio.
 - In turns, it decreases the information gain.
- Further, information gain is large when there are many distinct attribute values.
 - When many distinct values, split information is also a large value.
 - This way split information reduces the value of gain ratio, thus resulting a balanced value for information gain.
- Like information gain (in ID3), the attribute with the maximum gain ratio is selected as the splitting attribute in C4.5.

Calculation of β using Frequency Table

- The frequency table can be used to calculate the gain ratio for a given data set and an attribute.
- We have already learned the calculation of information gain using Frequency Table.
- To calculate gain ratio, in addition to information gain, we are also to calculate split information.
- This split information can be calculated from frequency table as follows.
- For each non-zero column sum say s_j contribute $|D_j|$ for the j -th column (i.e., the j -th value of the attribute). Thus the split information is

$$E_A^*(D) = - \sum_{j=1}^m \frac{s_j}{|D|} \log_2 \frac{s_j}{|D|}$$

If there are m -columns in the frequency table.

Practice:

Using Gain ratio as the measurement of splitting attributes, draw the decision trees for OPTH and EMP data sets. Give calculation of gain ratio at each node.

Summary of Decision Tree Induction Algorithms

- We have learned the building of a decision tree given a training data.
 - The decision tree is then used to classify a test data.
- For a given training data D , the important task is to build the decision tree so that:
 - All test data can be classified accurately
 - The tree is balanced and with as minimum depth as possible, thus the classification can be done at a faster rate.
- In order to build a decision tree, several algorithms have been proposed. These algorithms differ from the chosen splitting criteria, so that they satisfy the above mentioned objectives as well as the decision tree can be induced with minimum time complexity. We have studied three decision tree induction algorithms namely ID3, CART and C4.5. A summary of these three algorithms is presented in the following table.

Table 20.6

Algorithm	Splitting Criteria	Remark
ID3	<p>Information Gain</p> $\alpha(A, D) = E(D) - E_A(D)$ <p>Where</p> $E(D) = \text{Entropy of } D \text{ (a measure of uncertainty)} = -\sum_{i=1}^k p_i \log_2 p_i$ <p>where D is with set of k classes c_1, c_2, \dots, c_k and $p_i = \frac{ C_{i,D} }{ D }$; Here, $C_{i,D}$ is the set of tuples with class c_i in D.</p> $E_A(D) = \text{Weighted average entropy when } D \text{ is partitioned on the values of attribute A} = \sum_{j=1}^m \frac{ D_j }{ D } E(D_j)$ <p>Here, m denotes the distinct values of attribute A.</p>	<ul style="list-style-type: none"> The algorithm calculates $\alpha(A_i, D)$ for all A_i in D and choose that attribute which has maximum $\alpha(A_i, D)$. The algorithm can handle both categorical and numerical attributes. It favors splitting those attributes, which has a large number of distinct values.

Algorithm	Splitting Criteria	Remark
CART	<p>Gini Index</p> $\gamma(A, D) = G(D) - G_A(D)$ <p>where</p> $G(D) = \text{Gini index (a measure of impurity)}$ $= 1 - \sum_{i=1}^k p_i^2$ <p>Here, $p_i = \frac{ C_{i,D} }{ D }$ and D is with k number of classes and</p> $G_A(D) = \frac{ D_1 }{ D } G(D_1) + \frac{ D_2 }{ D } G(D_2),$ <p>when D is partitioned into two data sets D_1 and D_2 based on some values of attribute A.</p>	<ul style="list-style-type: none"> The algorithm calculates all binary partitions for all possible values of attribute A and choose that binary partition which has the minimum $\gamma(A, D)$. The algorithm is computationally very expensive when the attribute A has a large number of values.

Algorithm	Splitting Criteria	Remark
C4.5	<p>Gain Ratio</p> $\beta(A, D) = \frac{\alpha(A, D)}{E_A^*(D)}$ <p>where</p> $\alpha(A, D) = \text{Information gain of } D \text{ (same as in ID3, and)}$ $E_A^*(D) = \text{splitting information}$ $= - \sum_{j=1}^m \frac{ D_j }{ D } \log_2 \frac{ D_j }{ D }$ <p>when D is partitioned into D_1, D_2, \dots, D_m partitions corresponding to m distinct attribute values of A.</p>	<ul style="list-style-type: none"> The attribute A with maximum value of $\beta(A, D)$ is selected for splitting. Splitting information is a kind of normalization, so that it can check the biasness of information gain towards the choosing attributes with a large number of distinct values.

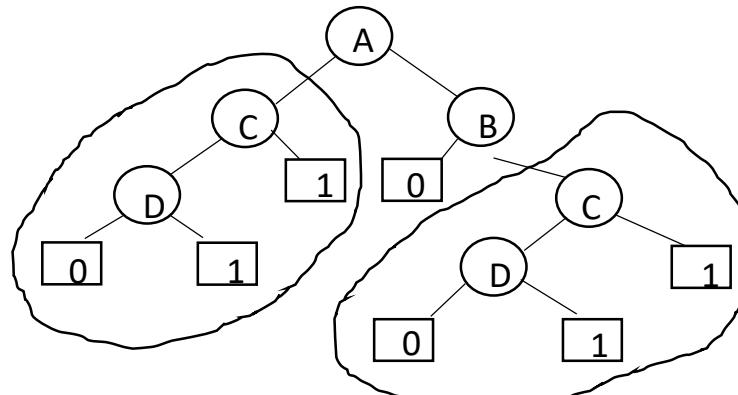
In addition to this, we also highlight few important characteristics of decision tree induction algorithms in the following.

Notes on Decision Tree Induction algorithms

1. **Optimal Decision Tree:** Finding an optimal decision tree is an NP-complete problem. Hence, decision tree induction algorithms **employ a heuristic based approach** to search for the best in a large search space. Majority of the algorithms follow a greedy, top-down recursive divide-and-conquer strategy to build decision trees.
2. **Missing data and noise:** Decision tree induction algorithms are quite robust to the data set with missing values and presence of noise. However, proper data pre-processing can be followed to nullify these discrepancies.
3. **Redundant Attributes:** The presence of redundant attributes does not adversely affect the accuracy of decision trees. It is observed that if an attribute is chosen for splitting, then another attribute which is redundant is unlikely to chosen for splitting.
4. **Computational complexity:** Decision tree induction algorithms are computationally inexpensive, in particular, when the sizes of training sets are large, Moreover, once a decision tree is known, classifying a test record is extremely fast, with a worst-case time complexity of $O(d)$, where d is the maximum depth of the tree.

Notes on Decision Tree Induction algorithms

5. **Data Fragmentation Problem:** Since the decision tree induction algorithms employ a top-down, recursive partitioning approach, the number of tuples becomes smaller as we traverse down the tree. At a time, the number of tuples may be too small to make a decision about the class representation, such a problem is known as the data fragmentation. To deal with this problem, further splitting can be stopped when the number of records falls below a certain threshold.
6. **Tree Pruning:** A sub-tree can replicate two or more times in a decision tree (see figure below). This makes a decision tree unambiguous to classify a test record. To avoid such a sub-tree replication problem, all sub-trees except one can be pruned from the tree.



Reference

- The detail material related to this lecture can be found in

Data Mining: Concepts and Techniques, (3rd Edn.), Jiawei Han, Micheline Kamber, Morgan Kaufmann, 2015.

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, Addison-Wesley, 2014

Any question?



Brain Computer Interaction

Performance Evaluation

Course Instructors

Dr. Sreeja S R

Assistant Professor

**Indian Institute of Information Technology
IIIT Sri City**

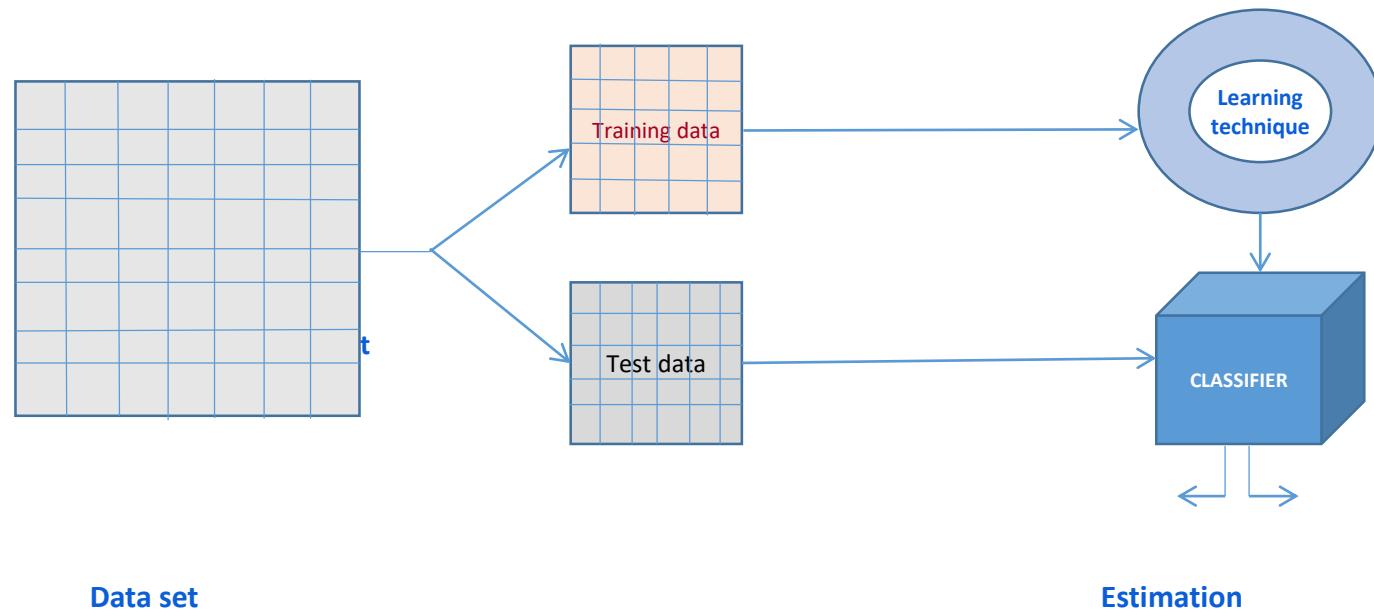
Introduction

- A classifier is used to predict an outcome of a test data
 - Such a prediction is useful in many applications
 - Business forecasting, cause-and-effect analysis, etc.
 - A number of classifiers have been evolved to support the activities.
 - Each has their own merits and demerits
- There is a need to estimate the accuracy and performance of the classifier with respect to few controlling parameters in data sensitivity
- As a task of sensitivity analysis, we have to focus on
 - Estimation strategy
 - Metrics for measuring accuracy
 - Metrics for measuring performance

Estimation Strategy

Planning for Estimation

- Using some “**training data**”, building a classifier based on certain principle is called “**learning a classifier**”.
- After building a classifier and before using it for classification of unseen instance, we have to validate it using some “**test data**”.
- Usually training data and test data are outsourced from a large pool of data already available.



Estimation Strategies

- Accuracy and performance measurement should follow a strategy. As the topic is important, many strategies have been advocated so far. Most widely used strategies are
 - Holdout method
 - Random subsampling
 - Cross-validation
 - Bootstrap approach

Holdout Method

- This is a basic concept of estimating a prediction.
 - Given a dataset, it is partitioned into **two disjoint sets** called **training set** and **testing set**.
 - Classifier is **learned** based on the training set and get **evaluated** with testing set.
 - Proportion of training and testing sets is at the discretion of analyst; typically **1:1 or 2:1**, and there is a **trade-off between these sizes** of these two sets.
 - If the training set is **too large**, then **model may be good enough**, but **estimation may be less reliable** due to small testing set and vice-versa.

Random Subsampling

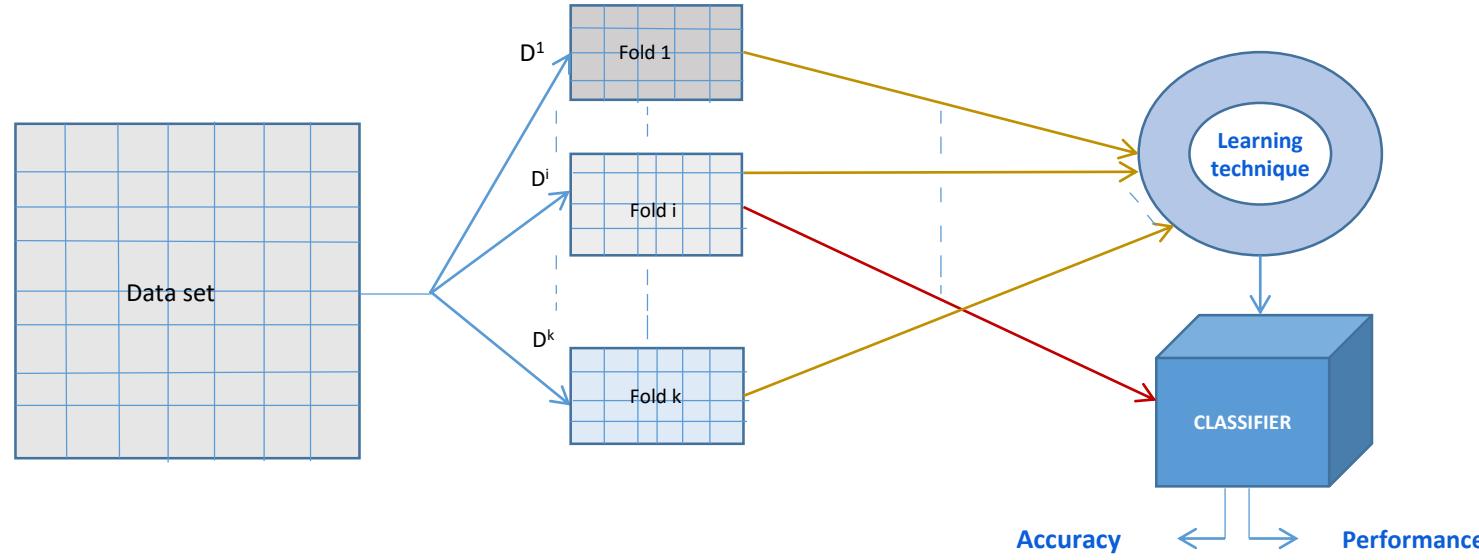
- It is a variation of Holdout method to overcome the drawback of over-presenting a class in one set thus under-presenting it in the other set and vice-versa.
- In this method, Holdout method is repeated k times, and in each time, two disjoint sets are chosen at random with a predefined sizes.
- Overall estimation is taken as the average of estimations obtained from each iteration.

Cross-Validation

- The main drawback of Random subsampling is, it does not have control over the number of times each tuple is used for training and testing.
- Cross-validation is proposed to overcome this problem.
- There are two variations in the cross-validation method.
 - k-fold cross-validation
 - N-fold cross-validation

k-fold Cross-Validation

- Dataset consisting of N tuples is divided into k (usually, 5 or 10) equal, mutually exclusive parts or folds (D_1, D_2, \dots, D_k), and if N is not divisible by k , then the last part will have fewer tuples than other ($k-1$) parts.
- A series of k runs is carried out with this decomposition, and in i^{th} iteration D_i is used as test data and other folds as training data
 - Thus, each tuple is used same number of times for training and once for testing.
- Overall estimate is taken as the average of estimates obtained from each iteration.



N-fold Cross-Validation

- In k -fold cross-validation method, $\frac{k-1}{N}$ part of the given data is used in training with k -tests.
- N -fold cross-validation is an [extreme case](#) of k -fold cross validation, often known as “Leave-one-out” cross-validation.
- Here, dataset is divided into as many folds as there are instances; thus, all most each tuple forming a training set, building N classifiers.
- In this method, therefore, N classifiers are built from $N-1$ instances, and each tuple is used to classify a single test instances.
- Test sets are mutually exclusive and effectively cover the entire set (in sequence). This is as if [trained by entire data as well as tested by entire data](#) set.
- Overall estimation is then averaged out of the results of N classifiers.

N-fold Cross-Validation : Issue

- So far the estimation of accuracy and performance of a classifier model is concerned, the *N*-fold cross-validation is comparable to the others we have just discussed.
- The drawback of *N*-fold cross validation strategy is that it is computationally expensive, as here we have to repeat the run *N* times; this is particularly true when data set is large.
- In practice, the method is extremely beneficial with very small data set only, where as much data as possible to need to be used to train a classifier.

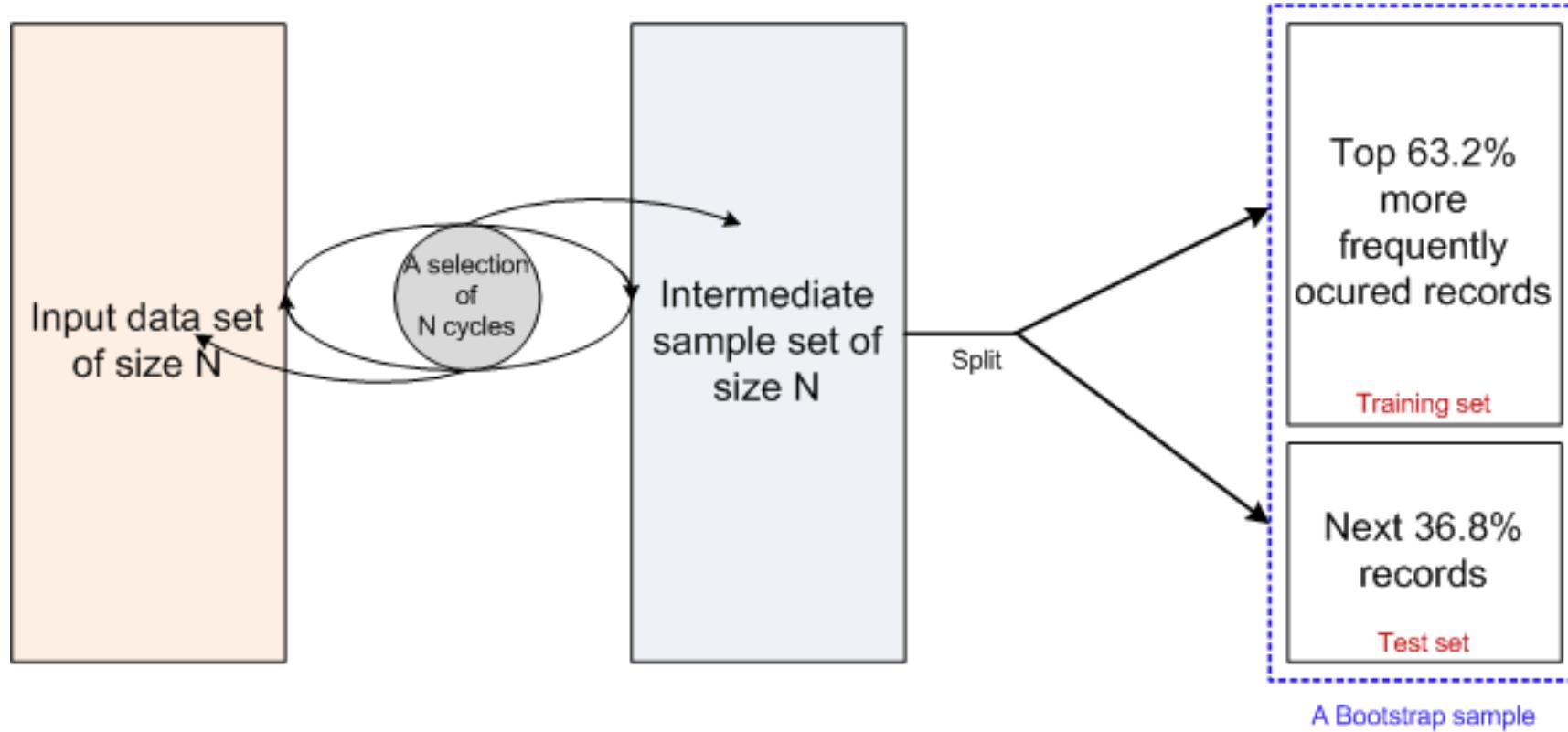
Bootstrap Method

- The Bootstrap method is a variation of **repeated version of Random sampling** method.
- The method suggests the **sampling of training records with replacement**.
 - Each time a record is selected for training set, is put back into the original pool of records, so that it is equally likely to be redrawn in the next run.
 - In other words, the Bootstrap method samples the given data set **uniformly with replacement**.
- The rational of having this strategy is that let some records be occur **more than once** in the samples of both training as well as testing.
 - What is the probability that a record will be selected more than once?

Bootstrap Method

- Suppose, we have given a data set of N records. The data set is sampled N times with replacement, resulting in a bootstrap sample (i.e., training set) of I samples.
 - Note that the entire runs are called a bootstrap sample in this method.
- There are certain chance (i.e., probability) that a particular tuple occurs **one or more** times in the training set
 - If they do not appear in the training set, then they will end up in the test set.
 - Each tuple has a probability of being selected $\frac{1}{N}$ (and the probability of not being selected is $(1 - \frac{1}{N})$).
 - We have to select N times, so the probability that a record will not be chosen during the whole run is $(1 - \frac{1}{N})^N$
 - Thus, the probability that a record is chosen by a bootstrap sample is $1 - (1 - \frac{1}{N})^N$
 - For a large value of N , it can be proved that $(1 - \frac{1}{N})^N \approx e^{-1}$
 - Thus, the probability that a record chosen in a bootstrap sample is $1 - e^{-1} = 0.632$

Bootstrap Method : Implication



- This is why, the Bootstrap method is also known as 0.632 bootstrap method

Accuracy Estimation

Accuracy Estimation

- We have learned how a classifier system can be tested. Next, we are to learn the metrics with which a classifier should be estimated.
- There are mainly two things to be measured for a given classifier
 - Accuracy
 - Performance
- **Accuracy estimation**
 - If N is the number of instances with which a classifier is tested and p is the number of correctly classified instances, the accuracy can be denoted as

$$\epsilon = \frac{p}{N}$$

- Also, we can say the **error rate** (i.e., is misclassification rate) denoted by $\bar{\epsilon}$ is denoted by

$$\bar{\epsilon} = 1 - \epsilon$$

•

Accuracy : True and Predictive

- Now, this accuracy may be **true** (or absolute) accuracy or **predicted** (or optimistic) accuracy.
- **True accuracy** of a classifier is the accuracy when the classifier is tested with **all possible unseen instances** in the given classification space.
 - However, the number of possible unseen instances is potentially very large (if it is not infinite)
 - For example, classifying a hand-written character
 - Hence, measuring the true accuracy beyond the dispute is impractical.
- **Predictive accuracy** of a classifier is an **accuracy estimation** for a given test data (which are mutually exclusive with training data).
 - If the predictive accuracy for test set is ϵ and if we test the classifier with a different test set it is very likely that a different accuracy would be obtained.
 - The predictive accuracy when estimated with a given test set it should be acceptable without any objection

Predictive Accuracy

Example 21.1 : Universality of predictive accuracy

- Consider a classifier model M^D developed with a training set D using an algorithm M.
- Two predictive accuracies when M^D is estimated with two different training sets T_1 and T_2 are

$$(M^D)_{T1} = 95\%$$

$$(M^D)_{T2} = 70\%$$

- Further, assume the size of T_1 and T_2 are
 - $|T_1| = 100$ records
 - $|T_2| = 5000$ records.
- Based on the above mentioned estimations, neither estimation is acceptable beyond doubt.

Error Estimation using Loss Functions

- Let T be a matrix comprising with N test tuples

$$\begin{bmatrix} X_1 & y_1 \\ X_2 & y_2 \\ \vdots & \vdots \\ X_N & y_N \end{bmatrix}_{N \times (n+1)}$$

where X_i ($i = 1, 2, \dots, N$) is the n -dimensional test tuples with associated outcome y_i .

- Suppose, corresponding to (X_i, y_i) , classifier produces the result (X_i, y'_i)
- Also, assume that $(y_i - y'_i)$ denotes a difference between y_i and y'_i (following certain difference (or similarity), (e.g., $(y_i - y'_i) = 0$, if there is a match else 1)
- The two loss functions measure the error between y_i (the actual value) and y'_i (the predicted value) are

$$\text{Absolute error: } |y_i - y'_i|$$

$$\text{Squared error: } |y_i - y'_i|^2$$

Error Estimation using Loss Functions

- Based on the two loss functions, the test error (rate) also called **generalization error**, is defined as the average loss over the test set T. The following two measures for test errors are

Mean Absolute Error (MAE):

$$\frac{\sum_{i=1}^N |y_i - y'_i|}{N}$$

Mean Squared Error(MSE):

$$\frac{\sum_{i=1}^N (y_i - y'_i)^2}{N}$$

- Note that, MSE aggregates the presence of outlier.
- In addition to the above, a relative error measurement is also known. In this measure, the error is measured relative to the mean value \tilde{y} calculated as the mean of y_i ($i = 1, 2, \dots, N$) of the training data say D. Two measures are

Relative Absolute Error (RAE):

$$\frac{\sum_{i=1}^N |y_i - y'_i|}{\sum_{i=1}^N |y_i - \tilde{y}|}$$

Relative Squared Error (RSE):

$$\frac{\sum_{i=1}^N (y_i - y'_i)^2}{\sum_{i=1}^N (y_i - \tilde{y})^2}$$

Statistical Estimation using Confidence Level

$$\tilde{\epsilon} = \epsilon \pm \tau_\alpha \times \sqrt{\epsilon(1-\epsilon)/N}$$

- A table of τ_α with different values of α can be obtained from any book on statistics. A small part of the same is given below.

α	0.5	0.7	0.8	0.9	0.95	0.98	0.99
τ_α	0.67	1.04	1.28	1.65	1.96	2.33	2.58

- Thus, given a confidence level α , we shall be able to know the value of τ_α and hence the true accuracy ($\tilde{\epsilon}$), if we have the value of the observed accuracy (ϵ).
- Thus, knowing a test data set of size N , it is possible to estimate the true accuracy!

Statistical Estimation using Confidence Level

Example 21.2: True accuracy from observed accuracy

A classifier is tested with a test set of size 100. Classifier predicts 80 test tuples correctly. We are to calculate the following.

- a) Observed accuracy
- b) Mean error rate
- c) Standard error
- d) True accuracy with confidence level 0.95.

Solution:

- a) The observed accuracy(ϵ) = $80/100 = 0.80$ So error (p) = 0.2
- b) Mean error rate = $p \times N = 0.2 \times 100 = 20$
- c) Standard error rate (σ) = $\sqrt{\epsilon(1-\epsilon)/N} = \sqrt{\frac{0.8 \times 0.2}{100}} = 0.04$
- d) $\tilde{\epsilon} = \epsilon \pm \tau_\alpha \times \sqrt{\epsilon(1-\epsilon)/N} = 0.8 \pm 0.04 \times 1.96 = 0.7216$ with $\tau_\alpha=1.96$ and $\alpha = 0.95$.

Statistical Estimation using Confidence Level

Note:

- Suppose, a classifier is tested k times with k different test sets. If ϵ_i denotes the predicted accuracy when tested with test set N_i in the i -th run ($1 \leq i \leq k$), then the overall predicted accuracy is

$$\epsilon = \sum_{i=1}^k \frac{\epsilon_i \times N_i}{\sum N_i}$$

Thus, ϵ is the weighted average of ϵ_i values. The standard error and true accuracy at a confidence α are

$$\text{Standard error} = \sqrt{\epsilon(1-\epsilon)/\sum_{i=1}^k N_i}$$

$$\text{True accuracy} = \epsilon \pm \sqrt{\frac{\epsilon(1-\epsilon)}{\sum_{i=1}^k N_i}} \times \tau_\alpha$$

Performance Estimation

Performance Estimation of a Classifier

- Predictive accuracy works fine, when the [classes are balanced](#)
 - That is, every class in the data set are equally important
- In fact, data sets with imbalanced class distributions are quite common in many real life applications
- When the classifier classified a test data set with imbalanced class distributions then, predictive accuracy on its own is not a reliable indicator of a classifier's effectiveness.

[Example 22.1: Effectiveness of Predictive Accuracy](#)

- Given a data set of stock markets, we are to classify them as “good” and “worst”. Suppose, in the data set, out of 100 entries, 98 belong to “good” class and only 2 are in “worst” class.
 - With this data set, if classifier’s predictive accuracy is 0.98, a very high value!
 - Here, there is a high chance that 2 “worst” stock markets may incorrectly be classified as “good”
 - On the other hand, if the predictive accuracy is 0.02, then none of the stock markets may be classified as “good”

Performance Estimation of a Classifier

- Thus, when the classifier classified a test data set with imbalanced class distributions, then predictive accuracy on its own is not a reliable indicator of a classifier's effectiveness.
- This necessitates an alternative metrics to judge the classifier.
- Before exploring them, we introduce the concept of **Confusion matrix**.

Confusion Matrix

- A confusion matrix for two classes (+, -) is shown below.

	C ₁	C ₂
C ₁	True positive	False negative
C ₂	False positive	True negative

	+	-
+	++	+-
-	-+	--

- There are four quadrants in the confusion matrix, which are symbolized as below.
 - True Positive (TP: f_{++})**: The number of instances that were positive (+) and correctly classified as positive (+v).
 - False Negative (FN: f_{+-})**: The number of instances that were positive (+) and incorrectly classified as negative (-).
 - False Positive (FP: f_{-+})**: The number of instances that were negative (-) and incorrectly classified as (+).
 - True Negative (TN: f_{--})**: The number of instances that were negative (-) and correctly classified as (-).

Confusion Matrix

Example 22.2: Confusion matrix

A classifier is built on a dataset regarding Good and Worst classes of stock markets. The model is then tested with a test set of 10000 unseen instances. The result is shown in the form of a confusion matrix. The result is self explanatory.

Class	Good	Worst	Total
Good	6954	46	7000
Worst	412	2588	3000
Total	7366	2634	10000

Predictive accuracy?

Performance Evaluation Metrics

- We now define a number of metrics for the measurement of a classifier.
 - In our discussion, we shall make the assumptions that there are only two classes: + (positive) and – (negative)
 - Nevertheless, the metrics can easily be extended to multi-class classifiers (with some modifications)
- **True Positive Rate (TPR):** It is defined as the fraction of the positive examples predicted correctly by the classifier.

$$TPR = \frac{TP}{P} = \frac{TP}{TP+FN} = \frac{f_{++}}{f_{++}+f_{+-}}$$

- This metric is also known as **Recall**, **Sensitivity** or **Hit rate**.
- **False Positive Rate (FPR):** It is defined as the fraction of negative examples classified as positive class by the classifier.

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} = \frac{f_{-+}}{f_{-+}+f_{--}}$$

- This metric is also known as **False Alarm Rate**.

Performance Evaluation Metrics

- **False Negative Rate (FNR):** It is defined as the fraction of positive examples classified as a negative class by the classifier.

$$FNR = \frac{FN}{P} = \frac{FN}{TP + FN} = \frac{f_{+-}}{f_{++} + f_{+-}}$$

- **True Negative Rate (TNR):** It is defined as the fraction of negative examples classified correctly by the classifier

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP} = \frac{f_{--}}{f_{--} + f_{-+}}$$

- This metric is also known as *Specificity*.

Performance Evaluation Metrics

- **Positive Predictive Value (PPV):** It is defined as the fraction of the positive examples classified as positive that are really positive

$$PPV = \frac{TP}{TP + FP} = \frac{f_{++}}{f_{++} + f_{-+}}$$

- It is also known as *Precision*.
- **F₁ Score (F₁):** Recall (r) and Precision (p) are two widely used metrics employed in analysis, where detection of one of the classes is considered more significant than the others.
 - It is defined in terms of (r or TPR) and (p or PPV) as follows.

$$\begin{aligned} F_1 &= \frac{2r.p}{r + p} = \frac{2TP}{2TP + FP + FN} \\ &= \frac{2f_{++}}{2f_{++} + f_{\mp} + f_{+-}} = \frac{2}{\frac{1}{r} + \frac{1}{p}} \end{aligned}$$

Predictive Accuracy (ε)

- It is defined as the fraction of the number of examples that are correctly classified by the classifier to the total number of instances.

$$\varepsilon = \frac{TP + TN}{P + N}$$

$$= \frac{TP + TN}{TP + FP + FN + TN}$$

$$= \frac{f_{++} + f_{--}}{f_{++} + f_{+-} + f_{-+} + f_{--}}$$

- This accuracy is equivalent to F_w with $w_1 = w_2 = w_3 = w_4 = 1$.

Error Rate ($\bar{\varepsilon}$)

- The error rate $\bar{\varepsilon}$ is defined as the fraction of the examples that are incorrectly classified.

$$\bar{\varepsilon} = \frac{FP + FN}{P + N}$$

$$= \frac{FP + FN}{TP + TN + FP + FN}$$

$$= \frac{f_{+-} + f_{-+}}{f_{++} + f_{+-} + f_{-+} + f_{--}}$$

Note

$$\bar{\varepsilon} = 1 - \varepsilon.$$

Accuracy, Sensitivity and Specificity

- Predictive accuracy (ε) can be expressed in terms of sensitivity and specificity.
- We can write

$$\varepsilon = \frac{TP + TN}{TP + FP + FN + TN}$$

$$= \frac{TP + TN}{P + N}$$

$$\varepsilon = \frac{TP}{P} \times \frac{P}{P + N} + \frac{TN}{N} \times \frac{N}{P + N}$$

Thus,

$$\varepsilon = \text{Sensitivity} \times \frac{P}{P+N} + \text{Specificity} \times \frac{N}{P+N}$$

Analysis with Performance Measurement Metrics

- Based on the various performance metrics, we can characterize a classifier.
- We do it in terms of TPR, FPR, Precision and Recall and Accuracy
- **Case 1: Perfect Classifier**

When every instance is **correctly** classified, it is called the **perfect classifier**. In this case, $TP = P$, $TN = N$ and CM is

$$TPR = \frac{P}{P} = 1$$

$$FPR = \frac{0}{N} = 0$$

$$Precision = \frac{P}{P} = 1$$

$$F_1 Score = \frac{2 \times 1}{1+1} = 1$$

$$Accuracy = \frac{P+N}{P+N} = 1$$

		Predicted Class	
		+	-
Actual class	+	P	0
	-	0	N

Analysis with Performance Measurement Metrics

- **Case 2: Worst Classifier**

When every instance is [wrongly](#) classified, it is called the [worst classifier](#). In this case, $TP = 0$, $TN = 0$ and the CM is

$$TPR = \frac{0}{P} = 0$$

$$FPR = \frac{N}{N} = 1$$

$$Precision = \frac{0}{N} = 0$$

F_1 Score = Not applicable

as $Recall + Precision = 0$

$$\text{Accuracy} = \frac{0}{P+N} = 0$$

		Predicted Class	
		+	-
Actual class	+	0	P
	-	N	0

Analysis with Performance Measurement Metrics

- **Case 3: Ultra-Liberal Classifier**

The classifier always predicts the + class correctly. Here, the False Negative (FN) and True Negative (TN) are zero. The CM is

$$TPR = \frac{P}{P} = 1$$

$$FPR = \frac{N}{N} = 1$$

$$Precision = \frac{P}{P+N}$$

$$F_1 Score = \frac{2P}{2P+N}$$

$$\text{Accuracy} = \frac{P}{P+N}$$

Actual class	Predicted Class	
	+	-
+	P	0
-	N	0

Analysis with Performance Measurement Metrics

- **Case 4: Ultra-Conservative Classifier**

This classifier always predicts the - class correctly. Here, the False Negative (FN) and True Negative (TN) are zero. The CM is

$$TPR = \frac{0}{P} = 0$$

$$FPR = \frac{0}{N} = 0$$

Precision = Not applicable
(as $TP + FP = 0$)

F₁ Score = Not applicable

$$\text{Accuracy} = \frac{N}{P+N}$$

		Predicted Class	
		+	-
Actual class	+	0	p
	-	0	N

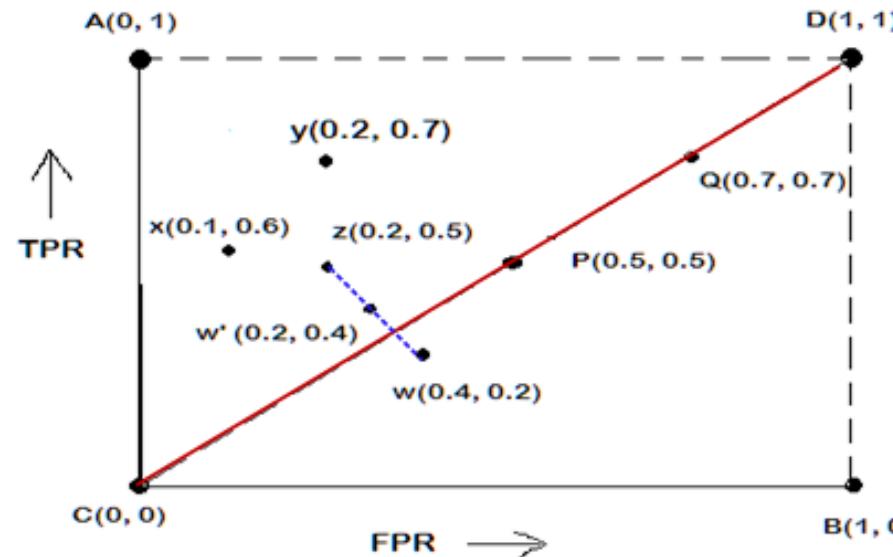
ROC Curves

ROC Curves

- ROC is an abbreviation of **Receiver Operating Characteristic** come from the signal detection theory, developed during World War 2 for analysis of radar images.
- In the context of classifier, ROC plot is a useful tool to study the behaviour of a classifier or **comparing two or more classifiers**.
- A ROC plot is **a two-dimensional graph**, where, X-axis represents FP rate (FPR) and Y-axis represents TP rate (TPR).
- Since, the values of FPR and TPR varies from 0 to 1 both inclusive, the two axes thus from 0 to 1 only.
- Each point (x, y) on the plot indicating that the FPR has value x and the TPR value y .

Interpretation of Different Points in ROC Plot

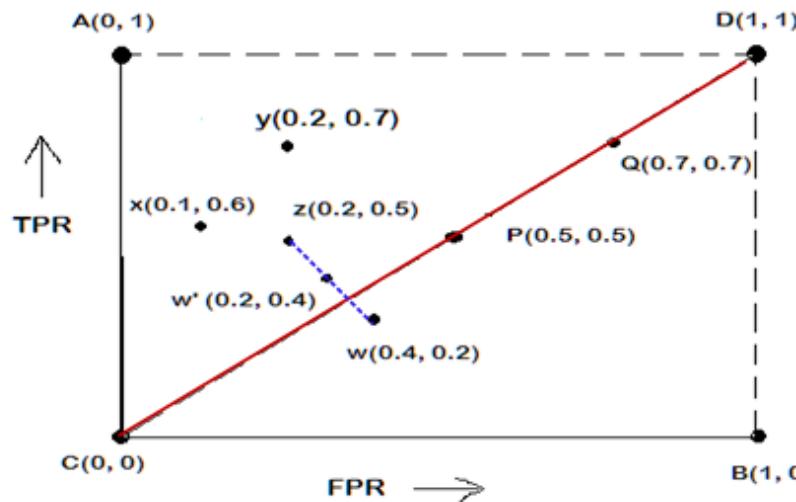
- Let us interpret the different points in the ROC plot.



- The four points (A, B, C, and D)
 - A: $\text{TPR} = 1, \text{FPR} = 0$, the ideal model, i.e., the **perfect classifier**, no false results
 - B: $\text{TPR} = 0, \text{FPR} = 1$, the **worst classifier**, not able to predict a single instance
 - C: $\text{TPR} = 0, \text{FPR} = 0$, the model predicts every instance to be a **Negative class**, i.e., it is an **ultra-conservative classifier**
 - D: $\text{TPR} = 1, \text{FPR} = 1$, the model predicts every instance to be a **Positive class**, i.e., it is an **ultra-liberal classifier**

Interpretation of Different Points in ROC Plot

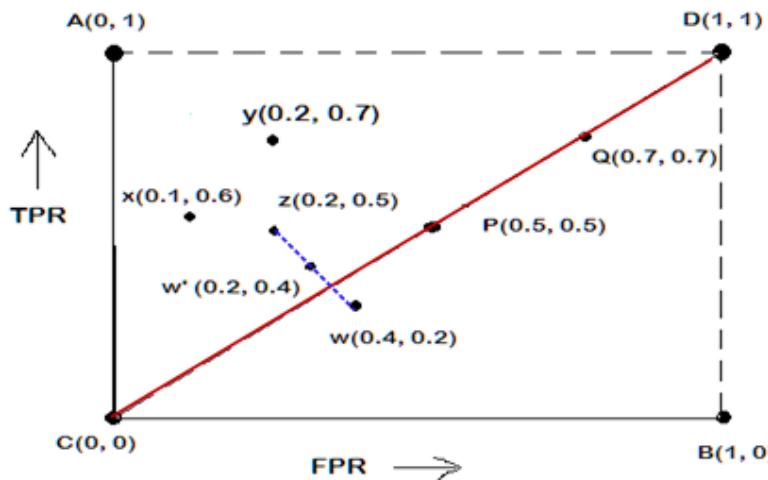
- Let us interpret the different points in the ROC plot.



- The points on diagonals
 - The diagonal line joining point C(0,0) and D(1,1) corresponds to random guessing
 - Random guessing means that a record is classified as positive (or negative) with a certain probability
 - Suppose, a test set containing N_+ positive and N_- negative instances. Suppose, the classifier guesses any instances with probability p
 - Thus, the random classifier is expected to correctly classify $p.N_+$ of the positive instances and $p.N_-$ of the negative instances
 - Hence, $TPR = FPR = p$
 - Since $TPR = FPR$, the random classifier results reside on the main diagonals

Interpretation of Different Points in ROC Plot

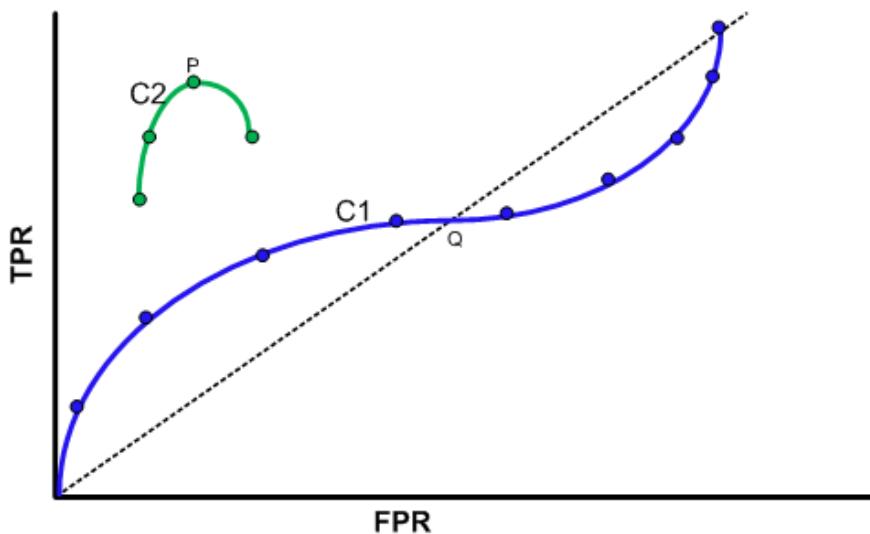
- Let us interpret the different points in the ROC plot.



- The points on the upper diagonal region
 - All points, which reside on upper-diagonal region are corresponding to classifiers “good” as their TPR is as good as FPR (i.e., FPRs are lower than TPRs)
 - Here, X is better than Z as X has higher TPR and lower FPR than Z.
 - If we compare X and Y, neither classifier is superior to the other

Tuning a Classifier through ROC Plot

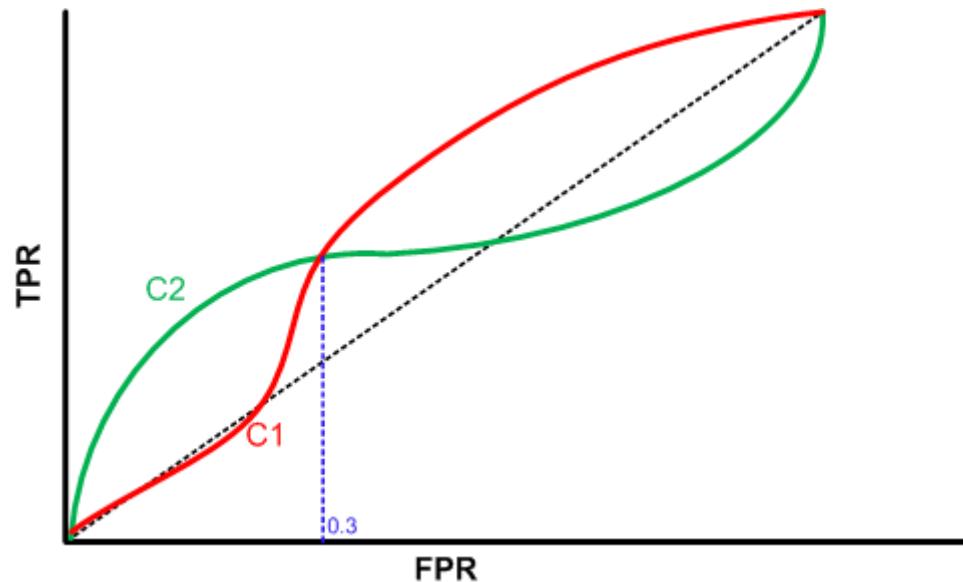
- Using ROC plot, we can compare two or more classifiers by their TPR and FPR values and this plot also depicts the trade-off between TPR and FPR of a classifier.



- Examining ROC curves can give insights into the best way of tuning parameters of classifier.
- For example, in the curve C2, the result is degraded after the point P. Similarly for the observation C1, beyond Q the settings are not acceptable.

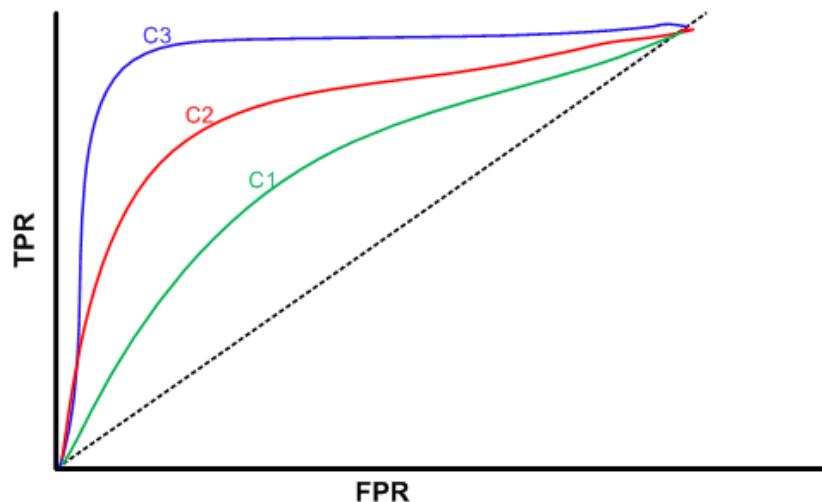
Comparing Classifiers through ROC Plot

- Two curves C1 and C2 are corresponding to the experiments to choose two classifiers with their parameters.
- Here, C1 is better than C2 when FPR is less than 0.3.
- However, C2 is better, when FPR is greater than 0.3.
- Clearly, neither of these two classifiers dominates the other.



Comparing Classifiers through ROC Plot

- We can use the concept of “area under curve” (AUC) as a better method to compare two or more classifiers.
- If a model is perfect, then its AUC = 1.
- If a model simply performs random guessing, then its AUC = 0.5
- A model that is strictly better than other, would have a larger value of AUC than the other.



- Here, C3 is best, and C2 is better than C1 as $AUC(C3) > AUC(C2) > AUC(C1)$.

Any question?



Brain Computer Interaction

Similarity Measures

Dr. Sreeja S R

Assistant Professor

**Indian Institute of Information Technology
IIIT Sri City**

Topics to be covered...

- Introduction to clustering
- Similarity and dissimilarity measures
- Clustering techniques
 - Partitioning algorithms
 - Hierarchical algorithms
 - Density-based algorithm

Introduction to Clustering

- Classification consists of assigning a class label to a set of unclassified cases.
- **Supervised Classification**
 - The set of possible classes is known in advance.
- **Unsupervised Classification**
 - Set of possible classes is not known. After classification we can try to assign a name to that class.
 - Unsupervised classification is called **clustering**.

Introduction to Clustering

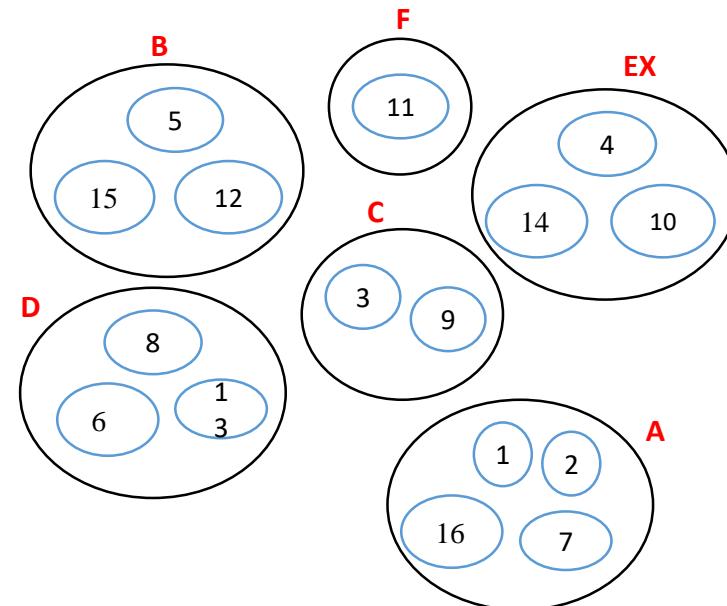
- Clustering is somewhat related to classification in the sense that in both cases data are grouped.
- However, there is a major difference between these two techniques.
- In order to understand the difference between the two, consider a sample dataset containing marks obtained by a set of students and corresponding grades as shown in Table 24.1.

Introduction to Clustering

Table 24.1: Tabulation of Marks

Roll No	Mark	Grade
1	80	A
2	70	A
3	55	C
4	91	EX
5	65	B
6	35	D
7	76	A
8	40	D
9	50	C
10	85	EX
11	25	F
12	60	B
13	45	D
14	95	EX
15	63	B
16	88	A

Figure 24.1: Group representation of dataset in Table 24.1

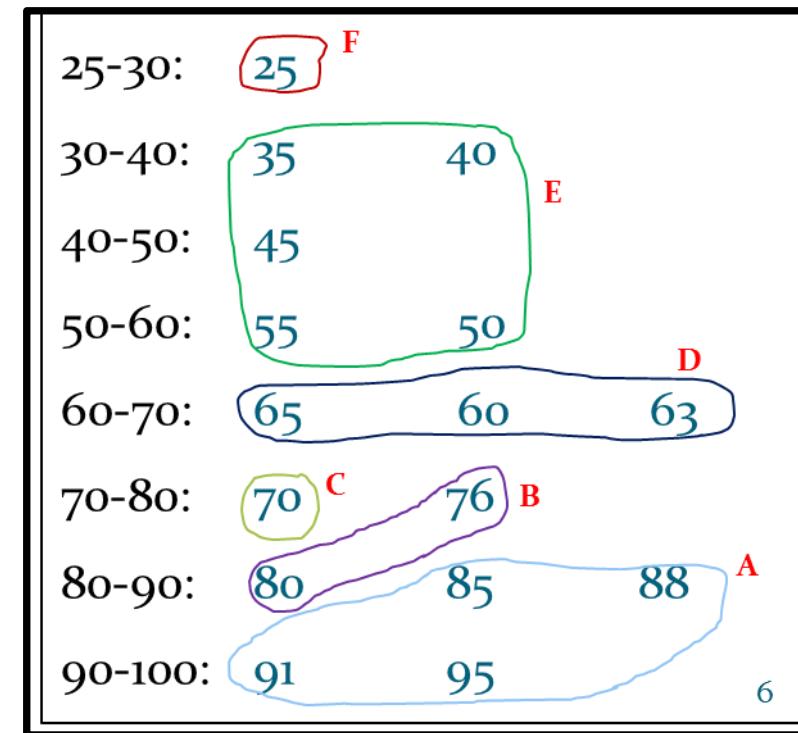


Introduction to Clustering

- It is evident that there is a simple mapping between Table 24.1 and Fig 24.1.
- The fact is that groups in Fig 24.1 are already predefined in Table 24.1. This is similar to classification, where we have given a dataset where **groups of data are predefined**.
- Consider another situation, where ‘Grade’ is not known, but we have to make a grouping.
- Put all the marks into a group if any other mark in that group does not exceed by 5 or more.
- This is similar to “**Relative grading**” concept and grade may range from A to Z.

Introduction to Clustering

- Figure 24.2 shows another grouping by means of another simple mapping, but the difference is **this mapping does not based on predefined classes**.
- In other words, this grouping is accomplished by finding **similarities between data according to characteristics found in the actual data**.
- Such a group making is called **clustering**.



Introduction to Clustering

Example 24.1 : The task of clustering

In order to elaborate the clustering task, consider the following dataset.

Table 24.2: Life Insurance database

Marital Status	Age	Income	Education	Number of children
Single	35	25000	Under Graduate	3
Married	25	15000	Graduate	1
Single	40	20000	Under Graduate	0
Divorced	20	30000	Post-Graduate	0
Divorced	25	20000	Under Graduate	3
Married	60	70000	Graduate	0
Married	30	90000	Post-Graduate	0
Married	45	60000	Graduate	5
Divorced	50	80000	Under Graduate	2

With certain similarity or likeliness defined, we can classify the records to one or group of more attributes (and thus mapping being non-trivial).

Introduction to Clustering

- Clustering has been used in many application domains:
 - Image analysis
 - Document retrieval
 - Machine learning, etc.
- When clustering is applied to real-world database, many problems may arise.

1. The (best) number of cluster is not known.

- There is no correct answer to a clustering problem.
- In fact, many answers may be found.
- The exact number of cluster required is not easy to determine.

Introduction to Clustering

2. There may not be any a priori knowledge concerning the clusters.
 - This is an issue that what data should be used for clustering.
 - Unlike classification, in clustering, we have no supervisory learning to aid the process.
 - Clustering can be viewed as similar to [unsupervised learning](#).

3. Interpreting the semantic meaning of each cluster may be difficult.
 - With classification, the labeling of classes is known ahead of time. In contrast, with clustering, this may not be the case.
 - Thus, when the clustering process is finished yielding a set of clusters, the exact meaning of each cluster may not be obvious.

Definition of Clustering Problem

Definition 24.1: Clustering

Given a database $D = \{t_1, t_2, \dots, t_n\}$ of n tuples, the clustering problem is to define a mapping $f : D \rightarrow C$, where each $t_i \in D$ is assigned to one cluster $c_i \in C$. Here, $C = \{c_1, c_2, \dots, c_k\}$ denotes a set of clusters.

- Solution to a clustering problem is devising a mapping formulation.
- The formulation behind such a mapping is to establish that a tuple within one cluster is **more like** tuples within that cluster and not similar to tuples outside it.

Definition of Clustering Problem

- Hence, mapping function f in Definition 24.1 may be explicitly stated as

$$f : D \rightarrow \{c_1, c_2, \dots, c_k\}$$

where i) each $t_i \in D$ is assigned to one cluster $c_i \in C$.

ii) for each cluster $c_i \in C$, and for all $t_{ip}, t_{iq} \in c_i$ and there exist $t_j \notin c_i$ such that

similarity (t_{ip}, t_{iq}) > similarity (t_{ip}, t_j) AND similarity (t_{iq}, t_j)

- In the field of cluster analysis, this **similarity** plays an important part.
- Now, we shall learn how similarity (this is also alternatively judged as “dissimilarity”) between any two data can be measured.

What is Similarity?

The quality or state of being similar; likeness; resemblance; as, a similarity of features.

Webster's Dictionary



Similarity is hard
to define, but...
*“We know it when
we see it”*

The real meaning
of similarity is a
philosophical
question. We will
take a more
pragmatic
approach.

Similarity and Dissimilarity Measures

- In clustering techniques, similarity (or dissimilarity) is an important measurement.
- Informally, **similarity** between two objects (e.g., two images, two documents, two records, etc.) is a numerical measure of the degree to which two objects are **alike**.
- The **dissimilarity** on the other hand, is another alternative (or opposite) measure of the degree to which two objects are **different**.
- Both similarity and dissimilarity also termed as **proximity**.
- Usually, similarity and dissimilarity are **non-negative numbers** and may range from **zero** (highly dissimilar (no similar)) to some finite/infinite value (highly similar (no dissimilar)).

Note:

- Frequently, the term **distance** is used as a synonym for dissimilarity
- In fact, it is used to refer as a special case of dissimilarity.

Proximity Measures: Single-Attribute

- Consider an object, which is defined by a single attribute A (e.g., length) and the attribute A has n -distinct values a_1, a_2, \dots, a_n .
- A data structure called “Dissimilarity matrix” is used to store a collection of proximities that are available for all pair of n attribute values.
 - In other words, the Dissimilarity matrix for an attribute A with n values is represented by an $n \times n$ matrix as shown below.

$$\begin{bmatrix} 0 & & & \\ p_{(2,1)} & 0 & & \\ p_{(3,1)} & p_{(3,2)} & 0 & \\ \vdots & \vdots & \vdots & \\ p_{(n,1)} & p_{(n,2)} & \dots & 0 \end{bmatrix}_{n \times n}$$

- Here, $p_{(i,j)}$ denotes the proximity measure between two objects with attribute values a_i and a_j .
- **Note:** The proximity measure is symmetric, that is, $p_{(i,j)} = p_{(j,i)}$

Proximity Calculation

- Proximity calculation to compute $p_{(i,j)}$ is different for different types of attributes according to NOIR topology.

Proximity calculation for Nominal attributes:

- For example, binary attribute, **Gender** = {Male, female} where **Male** is equivalent to **binary 1** and **female** is equivalent to **binary 0**.
- Similarity value is 1 if the two objects contains the same attribute value, while similarity value is 0 implies objects are not at all similar.

Object	Gender
R	Male
S	Female
L	Male

- Here, Similarity value let it be denoted by p , among different objects are as follows.

$$p(R, S) = 0$$

$$p(R, L) = 1$$

Note : In this case, if q denotes the **dissimilarity** between two objects i and j with single binary attributes, then $q_{(i,j)} = 1 - p_{(i,j)}$

Proximity Calculation

- Now, let us focus on how to calculate proximity measures between objects which are defined by two or more binary attributes.
- Suppose, the number of attributes be b . We can define the contingency table summarizing the different matches and mismatches between any two objects x and y , which are as follows.

Table 24.3: Contingency table with binary attributes

		Object y	
		1	0
Object x	1	f_{11}	f_{10}
	0	f_{01}	f_{00}

Here, f_{11} = the number of attributes where $x=1$ and $y=1$.

f_{10} = the number of attributes where $x=1$ and $y=0$.

f_{01} = the number of attributes where $x=0$ and $y=1$.

f_{00} = the number of attributes where $x=0$ and $y=0$.

Note : $f_{00} + f_{01} + f_{10} + f_{11} = b$, the total number of binary attributes.

Now, two cases may arise: symmetric and asymmetric binary attributes.

Similarity Measure with Symmetric Binary

- To measure the similarity between two objects defined by symmetric binary attributes using a measure called **symmetric binary coefficient** and denoted as \mathcal{S} and defined below

$$\mathcal{S} = \frac{\text{Number of matching attribute values}}{\text{Total number of attributes}}$$

or

$$\mathcal{S} = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}}$$

The **dissimilarity measure**, likewise can be denoted as \mathcal{D} and defined as

$$\mathcal{D} = \frac{\text{Number of mismatched attribute values}}{\text{Total number of attributes}}$$

or

$$\mathcal{D} = \frac{f_{01} + f_{10}}{f_{00} + f_{01} + f_{10} + f_{11}}$$

Note that, $\mathcal{D} = 1 - \mathcal{S}$

Similarity Measure with Symmetric Binary

Example 24.2: Proximity measures with symmetric binary attributes

Consider the following two dataset, where objects are defined with symmetric binary attributes.

Gender = {M, F}, Food = {V, N}, Caste = {H, M}, Education = {L, I},
Hobby = {T, C}, Job = {Y, N}

Object	Gender	Food	Caste	Education	Hobby	Job
Harry	M	V	M	L	C	N
Ram	M	N	M	I	T	N
Tomi	F	N	H	L	C	Y

$$\mathcal{S}(\text{Harry}, \text{Ram}) = \frac{1+2}{1+2+1+2} = 0.5$$

Proximity Measure with Asymmetric Binary

- Such a similarity measure between two objects defined by asymmetric binary attributes is done by **Jaccard Coefficient** and which is often symbolized by \mathcal{J} is given by the following equation

$$\mathcal{J} = \frac{\text{Number of matching presence}}{\text{Number of attributes not involved in 00 matching}}$$

or

$$\mathcal{J} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$

Proximity Measure with Asymmetric Binary

Example 24.3: Jaccard Coefficient

Consider the following two dataset.

Gender = {M, F}, Food = {V, N}, Caste = {H, M}, Education = {L, I},
Hobby = {T, C}, Job = {Y, N}

Calculate the Jaccard coefficient between Ram and Harry assuming that all binary attributes are asymmetric and for each pair values for an attribute, second one is more frequent than the first.

Object	Gender	Food	Caste	Education	Hobby	Job
Harry	M	V	M	L	C	N
Ram	M	N	M	I	T	N
Tomi	F	N	H	L	C	Y

$$\mathcal{J}(\text{Harry}, \text{Ram}) = \frac{2}{1+2+2} = 0.4$$

Note: $\mathcal{J}(\text{Ram}, \text{Tomi}) = 0$ and $\mathcal{J}(\text{Harry}, \text{Ram}) = \mathcal{J}(\text{Ram}, \text{Harry})$, etc.

Example 24.4:

Consider the following two dataset.

Gender = {M, F}, Food = {V, N}, Caste = {H, M}, Education = {L, I},
Hobby = {T, C}, Job = {Y, N}

Object	Gender	Food	Caste	Education	Hobby	Job
Harry	M	V	M	L	C	N
Ram	M	N	M	I	T	N
Tomi	F	N	H	L	C	Y



How you can calculate similarity if Gender, Hobby and Job are symmetric binary attributes and Food, Caste, Education are asymmetric binary attributes?

Obtain the similarity matrix with Jaccard coefficient of objects for the above, e.g.

$$\mathcal{J} = \begin{matrix} & H & R & T \\ H & 0 & 0 & 0 \\ R & \mathcal{J}(R, H) & 0 & 0 \\ T & \mathcal{J}(T, H) & \mathcal{J}(T, R) & 0 \end{matrix}$$

Proximity Measure with Categorical Attribute

- Binary attribute is a special kind of nominal attribute where the attribute has values with two states only.
- On the other hand, categorical attribute is another kind of nominal attribute where it has values with three or more states (e.g. color = {Red, Green, Blue}).
- If $s(x, y)$ denotes the similarity between two objects x and y , then

$$s(x, y) = \frac{\text{Number of matches}}{\text{Total number of attributes}}$$

and the dissimilarity $d(x, y)$ is

$$d(x, y) = \frac{\text{Number of mismatches}}{\text{Total number of attributes}}$$

- If m = number of matches and a = number of categorical attributes with which objects are defined as

$$s(x, y) = \frac{m}{a} \quad \text{and} \quad d(x, y) = \frac{a-m}{a}$$

Proximity Measure with Categorical Attribute

Example 24.4:

Object	Color	Position	Distance
1	R	L	L
2	B	C	M
3	G	R	M
4	R	L	H

The similarity matrix considering only color attribute is shown below

$$s = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

Dissimilarity matrix, $d = ?$

Obtain the dissimilarity matrix considering both the categorical attributes (i.e. color and position).

Proximity Measure with Ordinal Attribute

- Ordinal attribute is a special kind of categorical attribute, where the values of attribute follows a sequence (ordering) e.g. Grade = {Ex, A, B, C} where Ex > A >B >C.
- Suppose, A is an attribute of type ordinal and the set of values of $A = \{a_1, a_2, \dots, a_n\}$. Let n values of A are ordered in ascending order as $a_1 < a_2 < \dots < a_n$. Let i -th attribute value a_i be ranked as i , $i=1,2,\dots,n$.
- The normalized value of a_i can be expressed as

$$\hat{a}_i = \frac{i - 1}{n - 1}$$

- Thus, normalized values lie in the range [0..1].
- As a_i is a numerical value, the similarity measure, then can be calculated using any similarity measurement method for numerical attribute.
- For example, the similarity measure between two objects x and y with attribute values a_i and a_j , then can be expressed as

$$s(x, y) = \sqrt{(\hat{a}_i - \hat{a}_j)^2}$$

where \hat{a}_i and \hat{a}_j are the normalized values of \hat{a}_i and \hat{a}_j , respectively.

Proximity Measure with Ordinal Attribute

Example 24.5:

Consider the following set of records, where each record is defined by two ordinal attributes $\text{size} = \{\text{S, M, L}\}$ and $\text{Quality} = \{\text{Ex, A, B, C}\}$ such that $\text{S} < \text{M} < \text{L}$ and $\text{Ex} > \text{A} > \text{B} > \text{C}$.

Object	Size	Quality
A	S (0.0)	A (0.66)
B	L (1.0)	Ex (1.0)
C	L (1.0)	C (0.0)
D	M (0.5)	B (0.33)

- Normalized values are shown in brackets.
- Their similarity measures are shown in the similarity matrix below.

$$\begin{array}{ccccc} & & A & B & C & D \\ & & \hline A & [& 0 & 0 & 0 & 0 \\ B & [& 0 & 0 & 0 & 0 \\ C & [& ? & 0 & 0 & 0 \\ D & [& & 0 & 0 & 0 \end{array}$$

Find the dissimilarity matrix, when each object is defined by only one ordinal attribute say size (or quality).

Proximity Measure with Interval Scale

- The measure called **distance** is usually referred to estimate the similarity between two objects defined with interval-scaled attributes.
- We first present a generic formula to express distance d between two objects x and y in n -dimensional space. Suppose, x_i and y_i denote the values of i^{th} attribute of the objects x and y respectively.

$$d(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^r \right)^{\frac{1}{r}}$$

- Here, r is any integer value.
- This distance metric most popularly known as **Minkowski metric**.
- This distance measure follows some well-known properties. These are mentioned in the next slide.

Proximity Measure with Interval Scale

Properties of Minkowski metrics:

1. Non-negativity:

a. $d(x, y) \geq 0$ for all x and y

b. $d(x, y) = 0$ only if $x = y$. This is also called identity condition.

2. Symmetry:

$d(x, y) = d(y, x)$ for all x and y

This condition ensures that the order in which objects are considered is not important.

3. Transitivity:

$d(x, z) \leq d(x, y) + d(y, z)$ for all x, y and z .

- This condition has the interpretation that the least distance $d(x, z)$ between objects x and z is always less than or equal to the sum of the distance between the objects x and y , and between y and z .
- This property is also termed as Triangle Inequality.

Proximity Measure with Interval Scale

Depending on the value of r , the distance measure is renamed accordingly.

1. Manhattan distance (L_1 Norm: $r = 1$)

The Manhattan distance is expressed as

$$d = \sum_{i=1}^n |x_i - y_i|$$

where $|...|$ denotes the absolute value. This metric is also alternatively termed as **Taxicab metric, city-block metric**.

Example: $x = [7, 3, 5]$ and $y = [3, 2, 6]$.

The Manhattan distance is $|7 - 3| + |3 - 2| + |5 - 6| = 6$.

- As a special instance of Manhattan distance, when attribute values $\in [0, 1]$ is called **Hamming distance**.
- Alternatively, Hamming distance is the number of bits that are different between two objects that have only binary values (i.e. between two binary vectors).

Proximity Measure with Interval Scale

2. Euclidean Distance (L_2 Norm: $r = 2$)

This metric is same as Euclidean distance between any two points x and y in \mathcal{R}^n .

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Example: $x = [7, 3, 5]$ and $y = [3, 2, 6]$.

The Euclidean distance between x and y is

$$d(x, y) = \sqrt{(7 - 3)^2 + (3 - 2)^2 + (5 - 6)^2} = \sqrt{18} \approx 2.426$$

Proximity Measure with Interval Scale

3. Chebychev Distance (L_∞ Norm: $r \in \mathcal{R}$)

This metric is defined as

$$d(x, y) = \max_{\forall i} \{|x_i - y_i|\}$$

- We may clearly note the difference between Chebychev metric and Manhattan distance. That is, instead of summing up the absolute difference (in Manhattan distance), we simply take the maximum of the absolute differences (in Chebychev distance). Hence, $L_\infty < L_1$

Example: $x = [7, 3, 5]$ and $y = [3, 2, 6]$.

The Manhattan distance = $|7 - 3| + |3 - 2| + |5 - 6| = 6$.

The chebychev distance = Max $\{|7 - 3|, |3 - 2|, |5 - 6|\} = 4$.

Proximity Measure with Interval Scale

4. Other metrics:

a. Canberra metric:

$$d(x, y) = \sum_{i=1}^n \frac{|x_i - y_i|}{(|x_i| + |y_i|)^q}$$

- where q is a real number. Usually $q = 1$, because numerator of the ratio is always \leq denominator, the ratio ≤ 1 , that is, the sum is always bounded and small.
- If $q \neq 1$, it is called Fractional Canberra metric.
- If $q > 1$, the opposite relationship holds.

b. Hellinger metric:

$$d(x, y) = \sum_{i=1}^n (\sqrt{x_i} - \sqrt{y_i})^2$$

This metric is then used as either squared or transformed into an acceptable range [-1, +1] using the following transformations.

$$i. \quad d(x, y) = (1 - r(x, y))/2$$

$$ii. \quad d(x, y) = 1 - r(x, y)$$

Where $r(x, y)$ is correlation coefficient between x and y .

Proximity Measure for Ratio-Scale

The proximity between the objects with ratio-scaled variable can be carried with the following steps:

1. Apply the appropriate transformation to the data to bring it into a linear scale. (e.g. logarithmic transformation to data of the form $X = Ae^B$).
2. The transformed values can be treated as interval-scaled values. Any distance measure discussed for interval-scaled variable can be applied to measure the similarity.

Note:

There are two concerns on proximity measures:

- Normalization of the measured values.
- Intra-transformation from similarity to dissimilarity measure and vice-versa.

Proximity Measure for Ratio-Scale

Normalization:

- A major problem when using the similarity (or dissimilarity) measures (such as Euclidean distance) is that the large values frequently swamp the small ones.
- For example, consider the following data.

Make	Cost 1	Cost 2	Cost 3
X	2,00,000	70	10
Y	2,50,000	100	5

Here, the contribution of Cost 2 and Cost 3 is insignificant compared to Cost 1 so far the Euclidean distance is concerned.

- This problem can be avoided if we consider the normalized values of all numerical attributes.

Proximity Measure for Ratio-Scale

Intra-transformation:

- Transforming similarities to dissimilarities and vice-versa is also relatively straightforward.
- If the similarity (or dissimilarity) falls in the interval [0..1], the dissimilarity (or similarity) can be obtained as

$$\begin{aligned} d &= 1 - s \\ \text{or} \\ s &= 1 - d \end{aligned}$$

- Another approach is to define similarity as the negative of dissimilarity (or vice-versa).

Proximity Measure with Mixed Attributes

- The previous metrics on similarity measures assume that all the attributes were of the same type. Thus, a **general approach is needed when the attributes are of different types.**
- One straightforward approach is to compute the similarity between each attribute separately and then combine these attribute using a method that results in a similarity between 0 and 1.
- Typically, the overall similarity is defined as the average of all the individual attribute similarities.

Similarity Measure with Mixed Attributes

Example 24.6:

Consider the following set of objects. Obtain the similarity matrix.

Object	A (Binary)	B (Categorical)	C (Ordinal)	D (Numeric)	E (Numeric)
1	Y	R	X	475	10^8
2	N	R	A	10	10^{-2}
3	N	B	C	1000	10^5
4	Y	G	B	500	10^3
5	Y	B	A	80	1

$$\begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ ? & 0 & 0 & 0 & 0 \\ ? & ? & 0 & 0 & 0 \\ ? & ? & ? & 0 & 0 \\ ? & ? & ? & ? & 0 \end{matrix} \right] \end{matrix}$$

[For C: X>A>B>C]

How cosine similarity can be applied to this?

Non-Metric similarity

- In many applications (such as information retrieval) objects are complex and contains a large number of symbolic entities (such as keywords, phrases, etc.).
- To measure the distance between complex objects, it is often desirable to introduce a non-metric similarity function.
- Here, we discuss few such non-metric similarity measurements.

Cosine similarity

Suppose, x and y denote two vectors representing two complex objects. The cosine similarity denoted as $\cos(x, y)$ and defined as

$$\cos(x, y) = \frac{x \cdot y}{\|x\| \cdot \|y\|}$$

- where $x \cdot y$ denotes the vector dot product, namely $x \cdot y = \sum_{i=1}^n x_i \cdot y_i$ such that $x = [x_1, x_2, \dots, x_n]$ and $y = [y_1, y_2, \dots, y_n]$.
- $\|x\|$ and $\|y\|$ denote the Euclidean norms of vector x and y , respectively (essentially the length of vectors x and y), that is

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2} \text{ and } \|y\| = \sqrt{y_1^2 + y_2^2 + \dots + y_n^2}$$

Non-Metric Similarity

Example 24.7: Cosine Similarity

Suppose, we are given two documents with count of 10 words in each are shown in the form of vectors x and y as below.

$$x = [3, 2, 0, 5, 0, 0, 0, 2, 0, 0] \text{ and } y = [1, 0, 0, 0, 0, 0, 0, 1, 0, 2]$$

$$\begin{aligned} \text{Thus, } x \cdot y &= 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 \\ &= 5 \end{aligned}$$

$$\|x\| = \sqrt{3^2 + 2^2 + 0 + 5^2 + 0 + 0 + 0 + 2^2 + 0 + 0} = 6.48$$

$$\|y\| = \sqrt{1^2 + 0 + 0 + 0 + 0 + 0 + 0 + 1^2 + 0 + 2^2} = 2.24$$

$$\therefore \cos(x, y) = 0.31$$

Extended Jaccard Coefficient

The extended Jaccard coefficient is denoted as EJ and defined as

$$EJ = \frac{x \cdot y}{\|x\|^2 \cdot \|y\|^2 - x \cdot y}$$

- This is also alternatively termed as Tanimoto coefficient and can be used to measure like document similarity.

Compute Extended Jaccard coefficient (EJ) for the above example 24.7.

Pearson's Correlation

- The correlation between two objects x and y gives a measure of the linear relationship between the attributes of the objects.
- More precisely, Pearson's correlation coefficient between two objects x and y is defined in the following.

$$P(x, y) = \frac{S_{xy}}{S_x \cdot S_y}$$

where $S_{xy} = \text{covariance } (x, y) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$

$S_x = \text{Standard deviation } (x) = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$

$S_y = \text{Standard deviation } (y) = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (y_i - \bar{y})^2}$

$\bar{x} = \text{mean } (x) = \frac{1}{n} \sum_{i=1}^n x_i$

$\bar{y} = \text{mean } (y) = \frac{1}{n} \sum_{i=1}^n y_i$

and n is the number of attributes in x and y .

Note 1: Correlation is always in the range of -1 to 1. A correlation of 1(-1) means that x and y have a perfect positive (negative) linear relationship, that is, $x_i = a \cdot y_i + b$ for some a and b .

Example 24.8: Pearson's correlation

Calculate the Pearson's correlation of the two vectors x and y as given below.

$$x = [3, 6, 0, 3, 6]$$

$$y = [1, 2, 0, 1, 2]$$

Note: Vector components can be negative values as well.

Note:

If the correlation is 0, then there is no linear relationship between the attribute of the object.

Example 24.9: Non-linear correlation

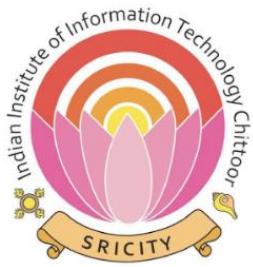
Verify that there is no linear relationship among attributes in the objects x and y given below.

$$x = [-3, -2, -1, 0, 1, 2, 3]$$

$$y = [9, 4, 1, 0, 1, 4, 9]$$

$P(x, y) = 0$, and also note $x_i = y_i^2$ for all attributes here.

Any question?



Brain Computer Interaction

Clustering techniques

Dr. Sreeja S R

Assistant Professor

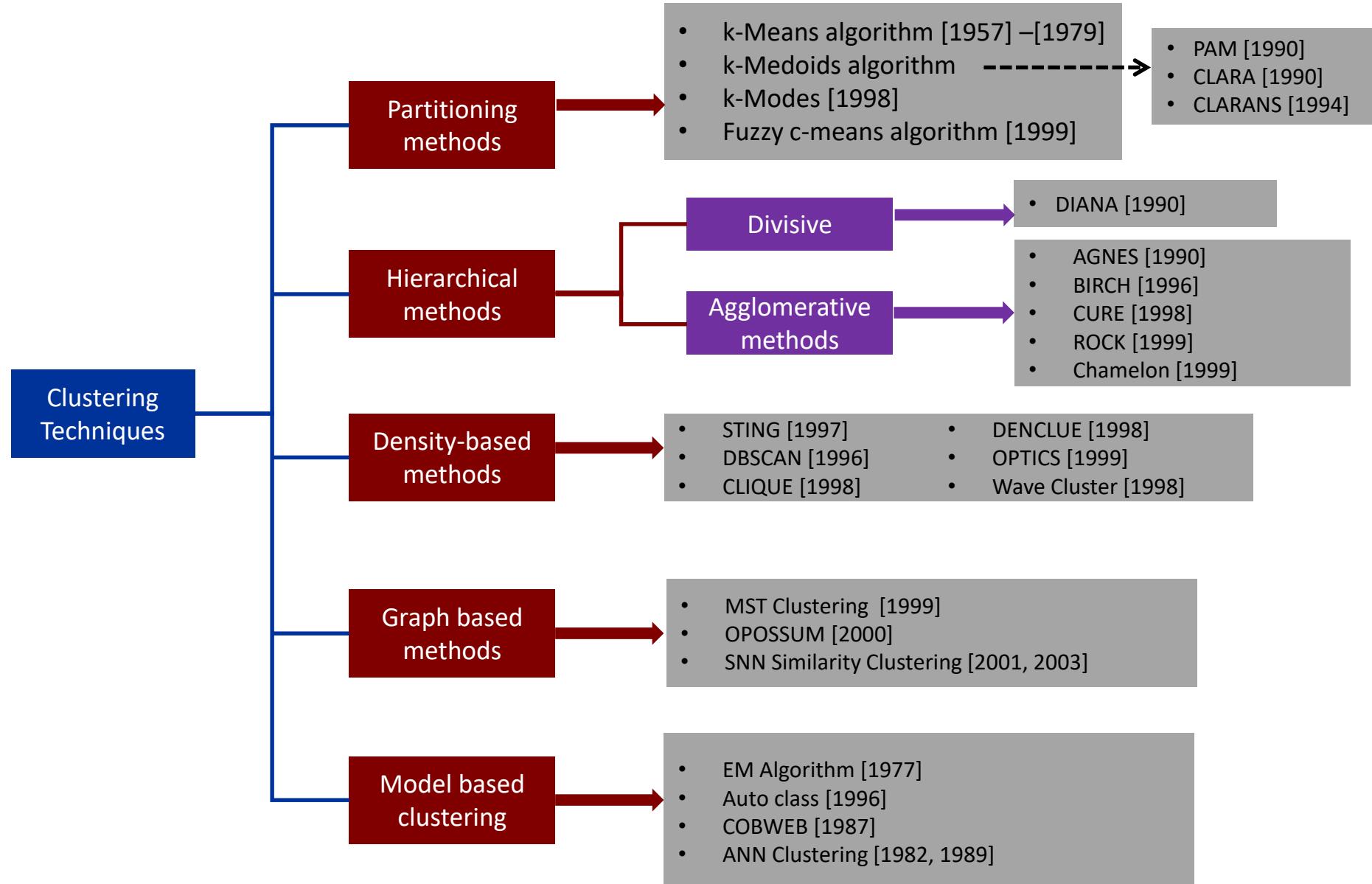
**Indian Institute of Information Technology
IIIT Sri City**

Topics to be covered...

- Introduction to clustering
- Similarity and dissimilarity measures
- Clustering techniques
- Partitioning algorithms
- Hierarchical algorithms
- Density-based algorithm

Clustering techniques

- Clustering has been studied extensively for more than 40 years and across many disciplines due to its broad applications.
- As a result, many clustering techniques have been reported in the literature.
- Let us categorize the clustering methods. In fact, it is difficult to provide a crisp categorization because many techniques overlap to each other in terms of clustering paradigms or features.
- A broad taxonomy of existing clustering methods is shown in the next slide.
- It is not possible to cover all the techniques in this lecture series. We emphasize on major techniques belong to partitioning and hierarchical algorithms.



k-Means Algorithm

- k-Means clustering algorithm proposed by J. Hartigan and M. A. Wong [1979].
- 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.

k-Means Algorithm

The algorithm can be stated as follows.

- 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.)

k-Means Algorithm

Algorithm 24.1: 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.
3. Compute the “cluster centers” of each cluster. These become the new cluster centroids.
4. Repeat step 2-3 until the convergence criterion is satisfied
5. Stop

k-Means Algorithm

Note:

- 1) Objects are defined in terms of set of attributes. $A = \{A_1, A_2, \dots, A_m\}$ 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.

Illustration of k-Means clustering algorithms

Table 24.1: 16 objects with two attributes A_1 and A_2 .

A_1	A_2
6.8	12.6
0.8	9.8
1.2	11.6
2.8	9.6
3.8	9.9
4.4	6.5
4.8	1.1
6.0	19.9
6.2	18.5
7.6	17.4
7.8	12.2
6.6	7.7
8.2	4.5
8.4	6.9
9.0	3.4
9.6	11.1

Fig 24.1: Plotting data of Table 24.1

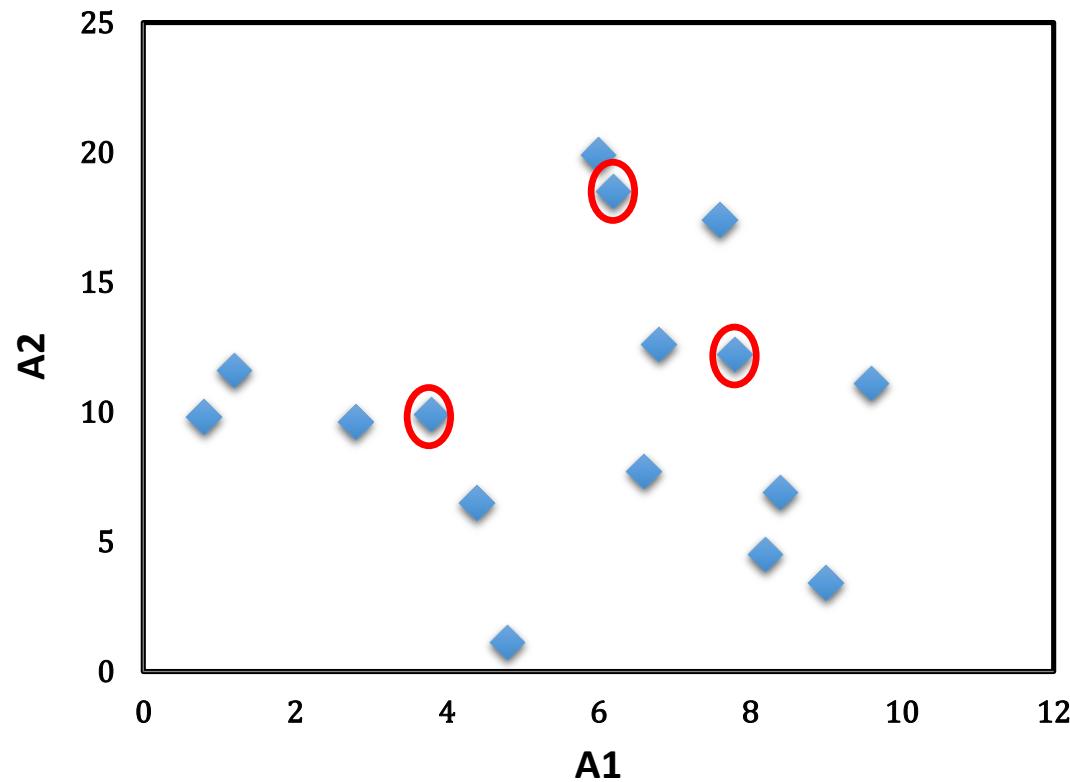


Illustration of k-Means clustering algorithms

- Suppose, $k=3$. Three objects are chosen at random shown as circled (see Fig 24.1). 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 24.2.
- Assignment of each object to the respective centroid is shown in the right-most column and the clustering so obtained is shown in Fig 24.2.

Illustration of k-Means clustering algorithms

Table 24.2: Distance calculation

A ₁	A ₂	d ₁	d ₂	d ₃	cluster
6.8	12.6	4.0	1.1	5.9	2
0.8	9.8	3.0	7.4	10.2	1
1.2	11.6	3.1	6.6	8.5	1
2.8	9.6	1.0	5.6	9.5	1
3.8	9.9	0.0	4.6	8.9	1
4.4	6.5	3.5	6.6	12.1	1
4.8	1.1	8.9	11.5	17.5	1
6.0	19.9	10.2	7.9	1.4	3
6.2	18.5	8.9	6.5	0.0	3
7.6	17.4	8.4	5.2	1.8	3
7.8	12.2	4.6	0.0	6.5	2
6.6	7.7	3.6	4.7	10.8	1
8.2	4.5	7.0	7.7	14.1	1
8.4	6.9	5.5	5.3	11.8	2
9.0	3.4	8.3	8.9	15.4	1
9.6	11.1	5.9	2.1	8.1	2

Fig 24.2: Initial cluster with respect to Table 24.2

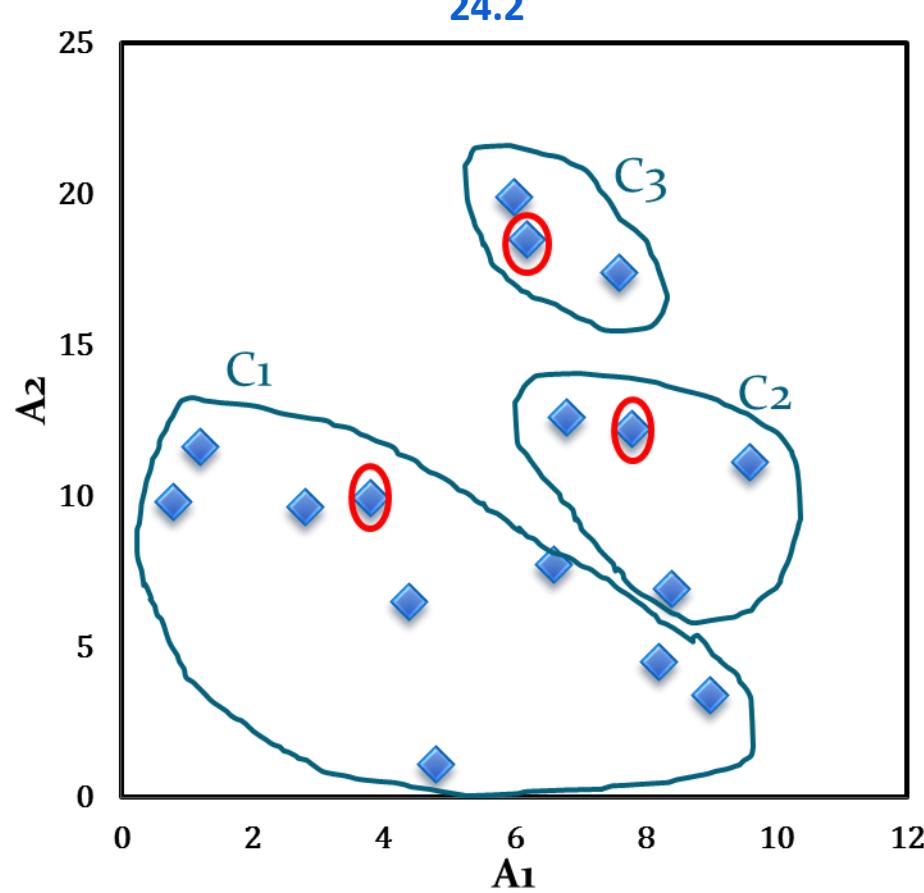


Illustration of k-Means clustering algorithms

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 24.3.

Calculation of new centroids

New Centroid	Objects	
	A_1	A_2
c_1	4.6	7.1
c_2	8.2	10.7
c_3	6.6	18.6

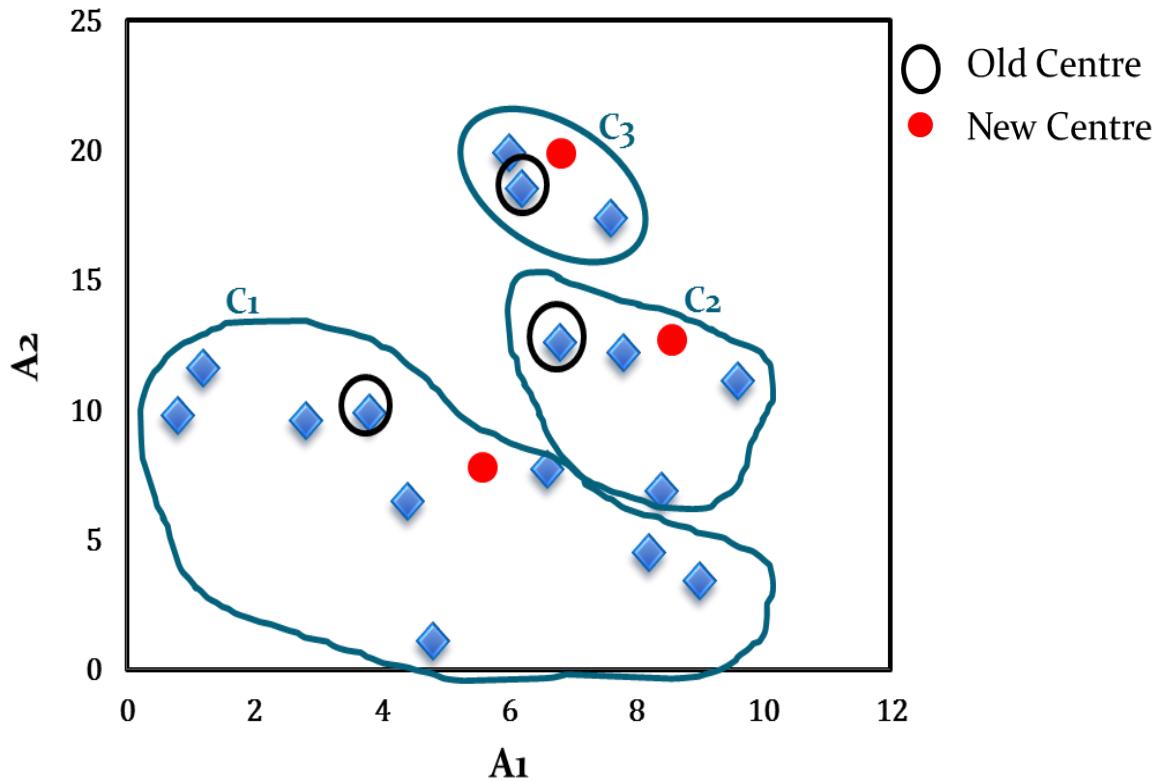


Fig 24.3: Initial cluster with new centroids

Illustration of k-Means clustering algorithms

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 24.4.

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

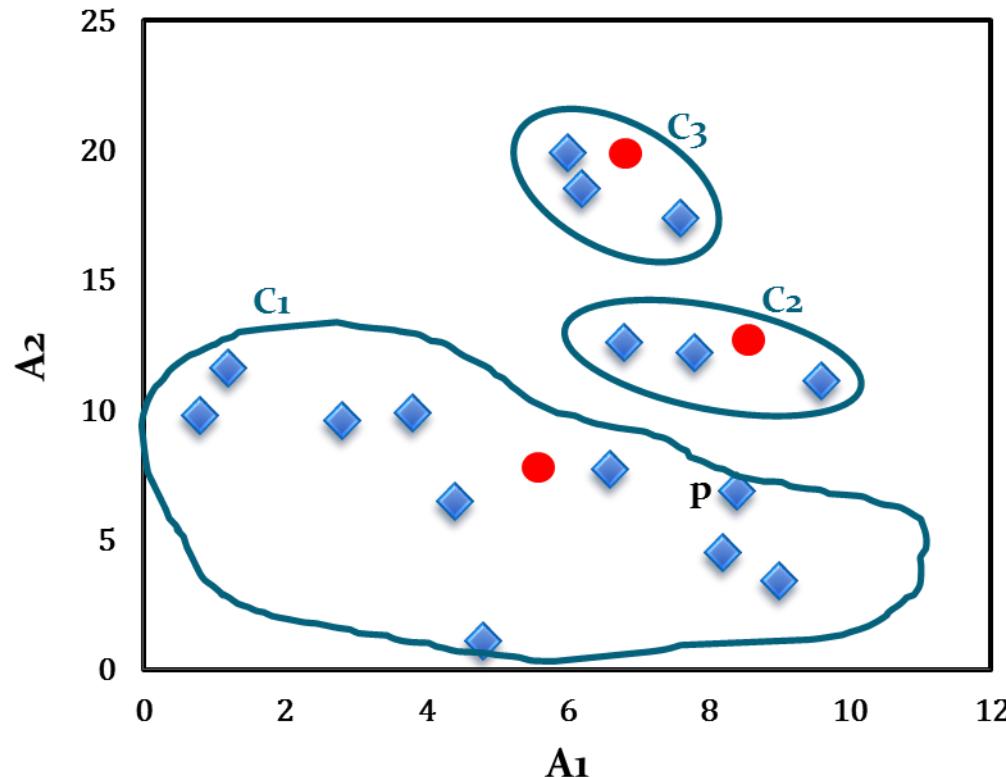


Fig 24.4: Cluster after first iteration

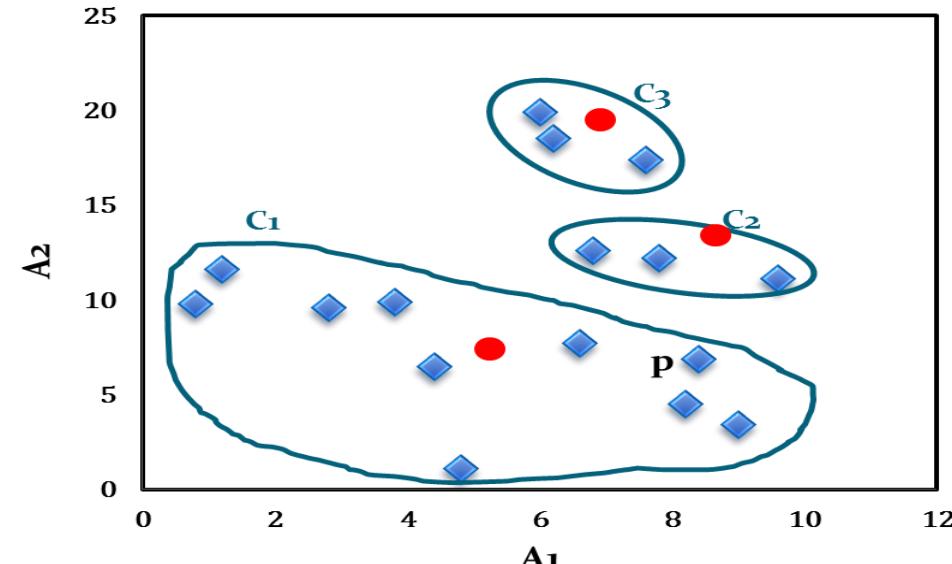
Illustration of k-Means clustering algorithms

- 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 24.5 is same as Fig 24.4.

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 24.5: Cluster after Second iteration



Comments on k-Means algorithm

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
- \mathcal{C}_i : the i -th cluster
- c_i : the centroid of cluster \mathcal{C}_i
- n_i : number of objects in the cluster \mathcal{C}_i
- c : denotes the centroid of all objects
- k : number of clusters

Comments on k-Means algorithm

1. Value of k:

- The k-means algorithm produces only one set of clusters, for which, user must specify the desired number, k of clusters.
- In fact, k should be the **best guess** on the number of clusters present in the given data. Choosing the best value of k for a given dataset is, therefore, an issue.
- We may not have an idea about the possible number of clusters for high dimensional data, and for data that are not scatter-plotted.
- Further, possible number of clusters is hidden or ambiguous in image, audio, video and multimedia clustering applications etc.
- There is no principled way to know what the value of k ought to be. We may try with successive value of k starting with 2.
- The process is stopped when two consecutive k values produce more-or-less identical results (with respect to some cluster quality estimation).
- Normally $k \ll n$ and there is heuristic to follow $k \approx \sqrt{n}$.

Comments on k-Means algorithm

Example 24.1: k versus cluster quality

- Usually, there is some objective function to be met as a goal of clustering. One such objective function is **sum-square-error** denoted by **SSE** and defined as

$$SSE = \sum_{i=1}^k \sum_{x \in C_i} (x - c_i)^2$$

- Here, $x - c_i$ denotes the error, if x is in cluster C_i with cluster centroid c_i .
- Usually, this error is measured as distance norms like L_1 , L_2 , L_3 or Cosine similarity, etc.

Comments on k-Means algorithm

Example 24.1: k versus cluster quality

- With reference to an arbitrary experiment, suppose the following results are obtained.

k	SSE
1	62.8
2	12.3
3	9.4
4	9.3
5	9.2
6	9.1
7	9.05
8	9.0

- With respect to this observation, we can choose the value of $k \approx 3$, as with this smallest value of k it gives reasonably good result.
- Note: If $k = n$, then SSE=0; However, the cluster is useless!

Comments on k-Means algorithm

2. Distance Measurement:

- To assign a point to the closest centroid, we need a proximity measure that should quantify the notion of “closest” for the objects under clustering.
- Usually Euclidean distance (L_2 norm) is the best measure when object points are defined in n-dimensional Euclidean space.
- Other measure namely cosine similarity is more appropriate when objects are of document type.
- Further, there may be other type of proximity measures that appropriate in the context of applications.
- For example, Manhattan distance (L_1 norm), Jaccard measure, etc.

Comments on k-Means algorithm

2. Distance Measurement:

Thus, in the context of different measures, the **sum-of-squared error** (i.e., objective function/convergence criteria) of a clustering can be stated as under.

Data in Euclidean space (L_2 norm):

$$SSE = \sum_{i=1}^k \sum_{x \in C_i} (c_i - x)^2$$

Data in Euclidean space (L_1 norm):

The Manhattan distance (L_1 norm) is used as a proximity measure, where the objective is to minimize the **sum-of-absolute error** denoted as **SAE** and defined as

$$SAE = \sum_{i=1}^k \sum_{x \in C_i} |c_i - x|$$

Comments on k-Means algorithm

3. Complexity analysis of k-Means algorithm

Let us analyse the time and space complexities of k-Means algorithm.

Time complexity:

The time complexity of the k-Means algorithm can be expressed as

$$T(n) = O(n \times m \times k \times l)$$

where n = number of objects

m = number of attributes in the object definition

k = number of clusters

l = number of iterations.

Thus, time requirement is a linear order of number of objects and the algorithm runs in a modest time if $k \ll n$ and $l \ll n$ (the iteration can be moderately controlled to check the value of l).

Comments on k-Means algorithm

3. Complexity analysis of k-Means algorithm

Space complexity: The storage complexity can be expressed as follows.

It requires $n \times m$ space to store the objects and $n \times k$ space to store the proximity measure from n objects to the centroids of k clusters.

Thus the total storage complexity is

$$S(n) = O(n \times (m + k))$$

That is, space requirement is in the linear order of n if $k \ll n$.

Comments on k-Means algorithm

4. Final comments:

Advantages:

- k-Means is simple and can be used for a wide variety of object types.
- It is also efficient both from storage requirement and execution time point of views. By saving distance information from one iteration to the next, the actual number of distance calculations, that must be made can be reduced (specially, as it reaches towards the termination).

Limitations:

- The k-Means is not suitable for all types of data. For example, k-Means does not work on categorical data because mean cannot be defined.
- k-means finds a local optima and may actually minimize the global optimum.

Comments on k-Means algorithm

4. Final comments:

Limitations :

- k-means has trouble clustering data that contains outliers. When the SSE is used as objective function, outliers can unduly influence the cluster that are produced. More precisely, in the presence of outliers, the cluster centroids, in fact, not truly as representative as they would be otherwise. It also influence the SSE measure as well.
- k-Means algorithm cannot handle non-globular clusters, clusters of different sizes and densities.
- k-Means algorithm not really beyond the scalability issue (and not so practical for large databases).

Comments on k-Means algorithm

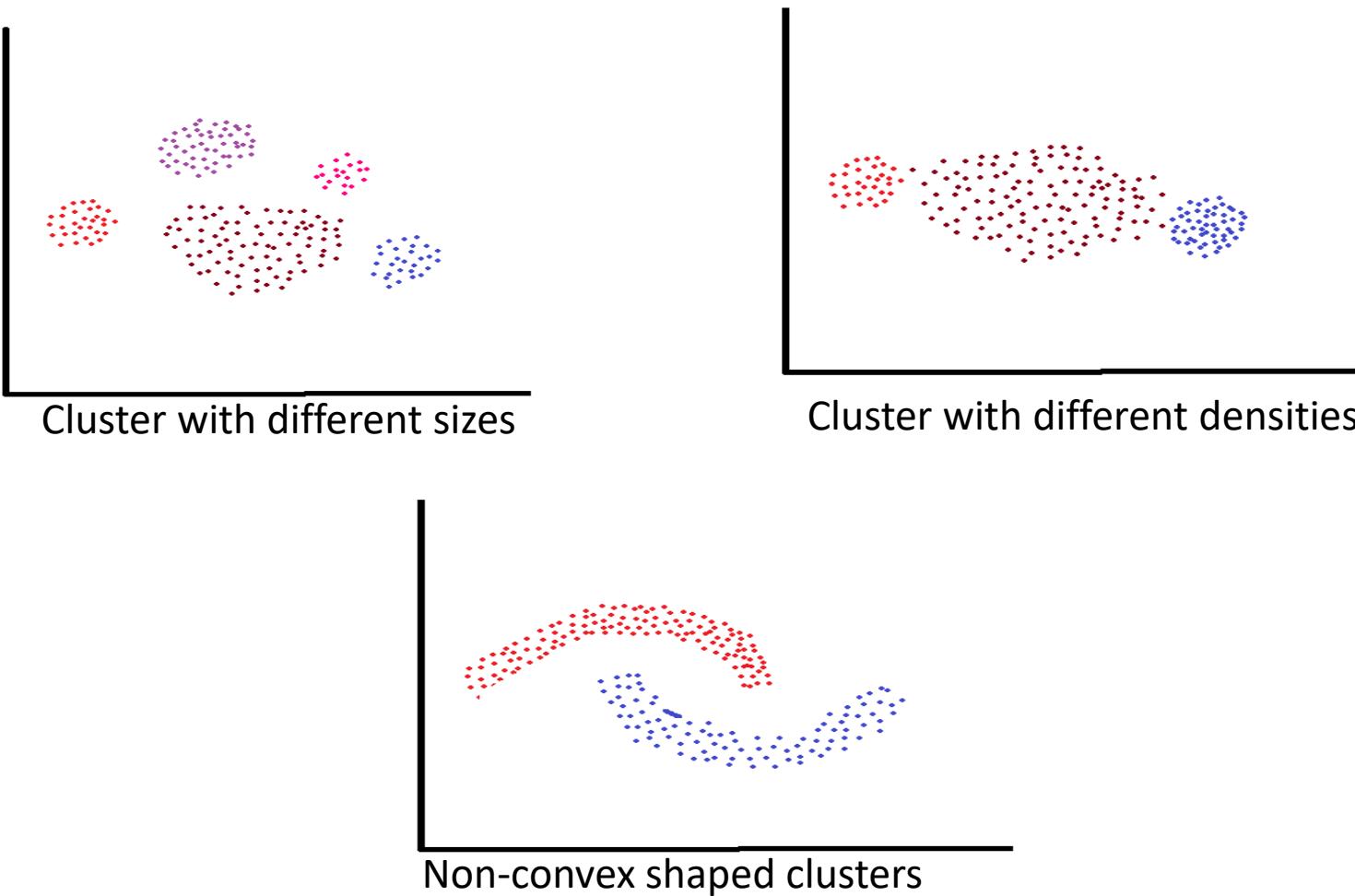


Fig 24.6: Some failure instance of k-Means algorithm

Different variants of k-means algorithm

There are quite a few variants of the k-Means algorithm. These can differ in the procedure of selecting the initial k means, the calculation of proximity and strategy for calculating cluster means. Another variants of k-means to cluster categorical data.

Few variant of k-Means algorithm includes

- Bisecting k-Means (addressing the issue of initial choice of cluster means).
 1. M. Steinbach, G. Karypis and V. Kumar “A comparison of document clustering techniques”, *Proceedings of KDD workshop on Text mining*, 2000.
- Mean of clusters (Proposing various strategies to define means and variants of means).
 - B. zhan “Generalised k-Harmonic means – Dynamic weighting of data in unsupervised learning”, *Technical report, HP Labs*, 2000.
 - A. D. Chaturvedi, P. E. Green, J. D. Carroll, “k-Modes clustering”, *Journal of classification*, Vol. 18, PP. 35-36, 2001.
 - D. Pelleg, A. Moore, “x-Means: Extending k-Means with efficient estimation of the number of clusters”, *17th International conference on Machine Learning*, 2000.

Different variants of k-means algorithm

- N. B. Karayiannis, M. M. Randolph, “Non-Euclidean c-Means clustering algorithm”, *Intelligent data analysis journal*, Vol 7(5), PP 405-425, 2003.
- V. J. Olivera, W. Pedrycy, “Advances in Fuzzy clustering and its applications”, Edited book. John Wiley [2007]. (Fuzzy c-Means algorithm).
- A. K. Jain and R. C. Bubes, “Algorithms for clustering Data”, Prentice Hall, 1988.
Online book at http://www.cse.msu.edu/~jain/clustering_Jain_Dubes.pdf
- A. K. Jain, M. N. Munty and P. J. Flynn, “Data clustering: A Review”, *ACM computing surveys*, 31(3), 264-323 [1999]. Also available online.

The k-Medoids algorithm

Now, we shall study a variant of partitioning algorithm called k-Medoids algorithm.

Motivation: We have learnt that the k-Means algorithm is sensitive to outliers because an object with an “extremely large value” may substantially distort the distribution. The effect is particularly exacerbated due to the use of the SSE (sum-of-squared error) objective function. The k-Medoids algorithm aims to diminish the effect of outliers.

Basic concepts:

- The basic concepts of this algorithm is to **select an object as a cluster center** (one representative object per cluster) instead of taking the mean value of the objects in a cluster (as in k-Means algorithm).
- We call this cluster representative as a **cluster medoid** or simply **medoid**.
 1. Initially, it selects a random set of k objects as the set of medoids.
 2. Then at each step, all objects from the set of objects, which are not currently medoids are examined one by one to see if they should be medoids.

The k-Medoids algorithm

- That is, the k-Medoids algorithm determines whether there is an object that should replace one of the current medoids.
- This is accomplished by looking all pair of medoid, non-medoid objects, and then choosing a pair that improves the objective function of clustering the best and exchange them.
- The sum-of-absolute error (SAE) function is used as the objective function.

$$SAE = \sum_{i=1}^k \sum_{x \in C_i, x \notin M \text{ and } c_m \in M} |x - c_m|$$

Where c_m denotes a medoid

M is the set of all medoids at any instant

x is an object belongs to set of non-medoid object, that is, x belongs to some cluster and is not a medoid. i.e. $x \in C_i, x \notin M$

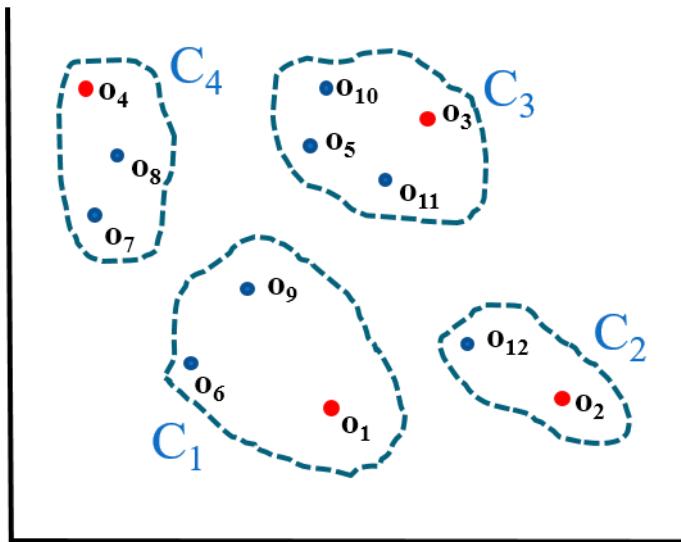
PAM (Partitioning around Medoids)

- For a given set of medoids, at any iteration, it select and exchange which has minimum SAE.
- The procedure terminates, if there is no change in SAE in successive iteration (i.e. there is no change in medoid).
- This k-Medoids algorithm is also known as PAM (Partitioning around Medoids).

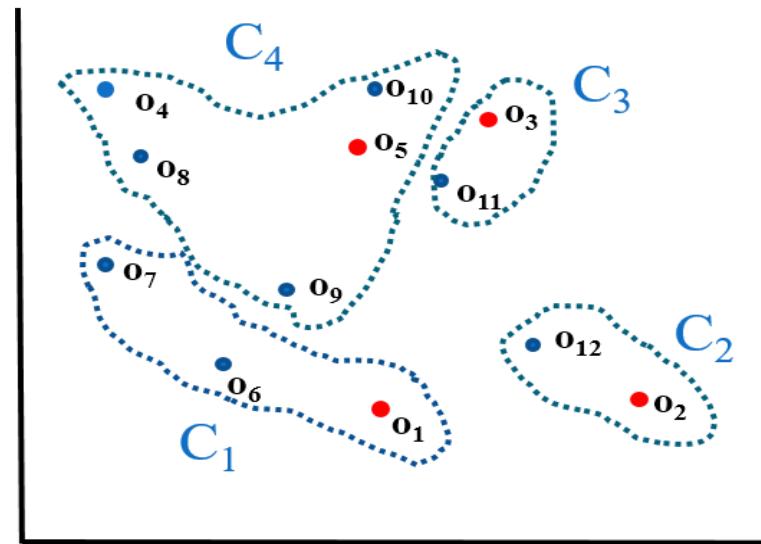
Illustration of PAM

- Suppose, there are set of 12 objects $O(o_1, o_2, \dots, o_{12})$ and we are to cluster them into four clusters. At any instant, the four cluster C_1, C_2, C_3 and C_4 are shown in Fig. 24.7 (a). Also assume that o_1, o_2, o_3 , and o_4 are the medoids in the clusters C_1, C_2, C_3 and C_4 , respectively. For this clustering we can calculate SAE.
- There are many ways to choose a non-medoid object to be replaced by any one medoid object. Out of these, suppose, if o_5 is considered as candidate medoid instead of o_4 , then it gives the lowest SAE. Thus, the new set of medoids would be o_1, o_2, o_3 , and o_5 . The new cluster is shown in Fig 24.7 (b).

PAM (Partitioning around Medoids)



(a) Cluster with o_1 , o_2 , o_3 , and o_4 as medoids



(b) Cluster after swapping o_4 and o_5 (o_5 becomes the new medoid).

Fig 24.7: Illustration of PAM

PAM (Partitioning around Medoids)

PAM algorithm is thus a procedure of iterative selection of medoids and it is precisely stated in Algorithm 24.2.

Algorithm 24.2: PAM

Input: Database of objects D.

k, the number of desired clusters.

Output: Set of k clusters

Steps:

1. Arbitrarily select k medoids from D.
2. **For** each object o_i not a medoid **do**
3. **For** each medoid o_j **do**
4. Let $M = \{o_1, o_2, \dots, o_{i-1}, o_i, o_{i+1}, o_k\}$ //Set of current medoids
 $M' = \{o_1, o_2, \dots, o_{j-1}, o_j, o_{j+1}, o_k\}$ //set of medoids but swap with non-medoids o_j
5. Calculate $\text{cost}(o_i, o_j) = SAE|_M - SAE_{M'}$,
6. **End** of 2 for loop

PAM (Partitioning around Medoids)

Algorithm 24.2: PAM

7. Find o_i, o_j for which the $\text{cost}(o_i, o_j)$ is the smallest.
8. Replace o_i with o_j and accordingly update the set M .
9. Repeat step 2 - step 8 until $\text{cost}(o_i, o_i) \leq 0$.
10. Return the cluster with M as the set of cluster centers.
11. Stop

Comments on PAM

1. Comparing k-Means with k-Medoids:

- Both algorithms needs to fix k , the number of cluster prior to the algorithms. Also, both algorithm arbitrarily choose the initial cluster centroids.
- The k-Medoid method is more robust than k-Means in the presence of outliers, because a medoid is less influenced by outliers than a mean.

2. Time complexity of PAM:

- For each iteration, PAM consider $k(n - k)$ pairs of object o_i, o_j for which a cost $\text{cost}(o_i, o_j)$ determines. Calculating the cost during each iteration requires that the cost be calculated for all other non-medoids o_j . There are $n - k$ of these. Thus, the total time complexity per iteration is $n(n - k)^2$. The total number of iterations may be quite large.

3. Applicability of PAM:

- PAM does not scale well to large database because of its computation complexity.

Other variants of k-Medoids algorithms

- There are some variants of PAM that are targeted mainly large datasets are CLARA (Clustering LARge Applications) and CLARANS (Clustering Large Applications based upon RANdomized Search), it is an improvement of CLARA.

References:

For PAM and CLARA:

- L. kaufman and P. J. Rousseeuw, “Finding Groups in Data: An introduction to cluster analysis”, John and Wiley, 1990.

For CLARANS:

- R. Ng and J. Han, “Efficient and effective clustering method for spatial Data mining”, Proceeding very large databases [VLDB-94], 1994.

Any question?