GATE Data Science and AI Study Materials

Machine Learning

by Piyush Wairale

Instructions:

- Kindly go through the lectures/videos on our website www.piyushwairale.com
- Read this study material carefully and make your own handwritten short notes. (Short notes must not be more than 5-6 pages)
- Attempt the mock tests available on portal.
- Revise this material at least 5 times and once you have prepared your short notes, then revise your short notes twice a week
- If you are not able to understand any topic or required a detailed explanation and if there are any typos or mistake in study materials. Mail me at piyushwairale100@gmail.com

Contents

LinkedIn

Youtube Channel

Instagram

Telegram Group

Facebook

Download Andriod App

1 Introduction to Machine Learning

Defination

- Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.
- Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data, or both.
- The field of study known as machine learning is concerned with the question of how to construct computer programs that automatically improve with experience

Defination of Learning

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks T, as measured by P, improves with experience E.
A computer program which learns from experience is called a machine learning program or simply a learning program. Such a program is sometimes also referred to as a learner.

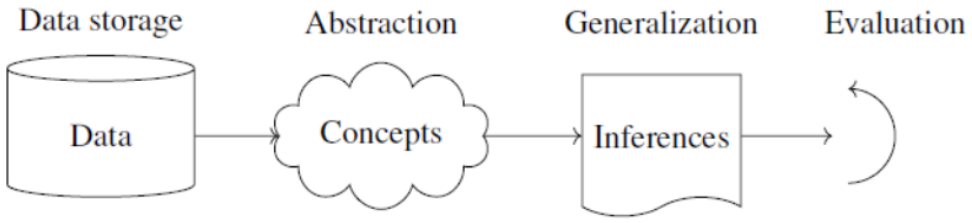
Examples

1. Handwriting recognition learning problem
    - Task T: Recognising and classifying handwritten words within images
    - Performance P: Percent of words correctly classified
    - Training experience E: A dataset of handwritten words with given classifications
2. A robot driving learning problem
    - Task T: Driving on highways using vision sensors
    - Performance measure P: Average distance traveled before an error
    - training experience: A sequence of images and steering commands recorded while observing a human driver
3. A chess learning problem.
    - Task T: Playing chess
    - Performance measure P: Percent of games won against opponents

- Training experience E: Playing practice games against itself as measured by P, improves with experience E.

How Machine Learn?

The learning process, whether by a human or a machine, can be divided into four components, namely, data storage, abstraction, generalization and evaluation.



1.1 Introduction to Data in Machine Learning

What is data?

It can be any unprocessed fact, value, text, sound, or picture that is not being interpreted and analyzed. Data is the most important part of all Data Analytics, Machine Learning, and Artificial Intelligence.

Without data, we can't train any model and all modern research and automation will go in vain.

- Data is a crucial component in the field of Machine Learning. It refers to the set of observations or measurements that can be used to train a machine-learning model.
- The quality and quantity of data available for training and testing play a significant role in determining the performance of a machine-learning model.
- Data can be in various forms such as numerical, categorical, or time-series data, and can come from various sources such as databases, spreadsheets, or APIs.
- Machine learning algorithms use data to learn patterns and relationships between input variables and target outputs, which can then be used for prediction or classification tasks.

Understanding data

Since an important component of the machine learning process is data storage, we briefly consider in this section the different types and forms of data that are encountered in the machine learning process.

Unit of observation

By a unit of observation we mean the smallest entity with measured properties of interest for a study.

Examples

- A person, an object or a thing
- A time point
- A geographic region
- A measurement

Sometimes, units of observation are combined to form units such as person-years.

Examples and features

Datasets that store the units of observation and their properties can be imagined as collections of data consisting of the following:

Examples

An "example" is an instance of the unit of observation for which properties have been recorded. An "example" is also referred to as an "instance", or "case" or "record." (It may be noted that the word "example" has been used here in a technical sense.)

Features

A "feature" is a recorded property or a characteristic of examples. It is also referred to as "attribute", or "variable" or "feature."

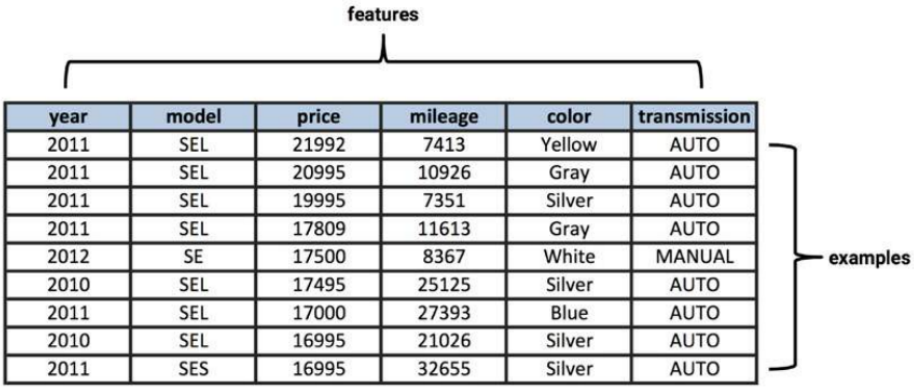
Examples for "examples" and "features"

1. Cancer detection

Consider the problem of developing an algorithm for detecting cancer. In this study we note the following.
(a) The units of observation are the patients.
(b) The examples are members of a sample of cancer patients.
(c) The following attributes of the patients may be chosen as the features:
• gender
• age
• blood pressure
• the findings of the pathology report after a biopsy

2. Pet selection

Suppose we want to predict the type of pet a person will choose.
(a) The units are the persons.
(b) The examples are members of a sample of persons who own pets
(c) The features might include age, home region, family income, etc. of persons who own pets.



year	model	price	mileage	color	transmission
2011	SEL	21992	7413	Yellow	AUTO
2011	SEL	20995	10926	Gray	AUTO
2011	SEL	19995	7351	Silver	AUTO
2011	SEL	17809	11613	Gray	AUTO
2012	SE	17500	8367	White	MANUAL
2010	SEL	17495	25125	Silver	AUTO
2011	SEL	17000	27393	Blue	AUTO
2010	SEL	16995	21026	Silver	AUTO
2011	SES	16995	32655	Silver	AUTO

Figure 1: Figure 1: Example for "examples" and "features" collected in a matrix format (data relates to automobiles and their features)

Type of Data

Data is typically divided into two types:

1. **Labeled data**
2. **Unlabeled data**

Labeled data includes a label or target variable that the model is trying to predict, whereas **unlabeled data** does not include a label or target variable. The data used in machine learning is typically numerical or categorical. Numerical data includes values that can be ordered and measured, such as age or income. Categorical data includes values that represent categories, such as gender or type of fruit.

- Data can be divided into training and testing sets.
- The training set is used to train the model, and the testing set is used to evaluate the performance of the model.

- It is important to ensure that the data is split in a random and representative way.
- Data preprocessing is an important step in the machine learning pipeline. This step can include cleaning and normalizing the data, handling missing values, and feature selection or engineering.

How do we split data in Machine Learning?

1. **Training Data:** The part of data we use to train our model. This is the data that your model actually sees(both input and output) and learns from.
2. **Validation Data:** The part of data that is used to do a frequent evaluation of the model, fit on the training dataset along with improving involved hyperparameters (initially set parameters before the model begins learning). This data plays its part when the model is actually training.
3. **Testing Data:** Once our model is completely trained, testing data provides an unbiased evaluation. When we feed in the inputs of Testing data, our model will predict some values(without seeing actual output).

   After prediction, we evaluate our model by comparing it with the actual output present in the testing data. This is how we evaluate and see how much our model has learned from the experiences feed in as training data, set at the time of training.

Different forms of data

1. **Numeric data** If a feature represents a characteristic measured in numbers, it is called a numeric feature.
2. **Categorical or nominal** A categorical feature is an attribute that can take on one of a limited, and usually fixed, number of possible values on the basis of some qualitative property. A categorical feature is also called a nominal feature.
3. **Ordinal data** This denotes a nominal variable with categories falling in an ordered list. Examples include clothing sizes such as small, medium, and large, or a measurement of customer satisfaction on a scale from "not at all happy" to "very happy."

Examples In the data given in Fig.1, the features "year", "price" and "mileage" are numeric and the features "model", "color" and "transmission" are categorical.

Properties of Data

- **Volume:** Scale of Data. With the growing world population and technology at exposure, huge data is being generated each and every millisecond.
- **Variety:** Different forms of data – healthcare, images, videos, audio clippings.
- **Velocity:** Rate of data streaming and generation.

- **Value:** Meaningfulness of data in terms of information that researchers can infer from it.
- **Veracity:** Certainty and correctness in data we are working on.
- **Viability:** The ability of data to be used and integrated into different systems and processes.
- Security: The measures taken to protect data from unauthorized access or manipulation.
- **Accessibility:** The ease of obtaining and utilizing data for decision-making purposes. Integrity: The accuracy and completeness of data over its entire lifecycle.
- **Usability:** The ease of use and interpretability of data for end-users.

1.2 Different types of learning

In general, machine learning algorithms can be classified into three types.

1. **Supervised learning**

   - Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.
   - In supervised learning, each example in the training set is a pair consisting of an input object (typically a vector) and an output value.
   - A supervised learning algorithm analyzes the training data and produces a function, which can be used for mapping new examples.
   - In the optimal case, the function will correctly determine the class labels for unseen instances.
   - Both **classification and regression problems** are supervised learning problems.
   - A wide range of supervised learning algorithms are available, each with its strengths and weaknesses. There is no single learning algorithm that works best on all supervised learning problems.
   - **Important Point** :A "supervised learning" is so called because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers (that is, the correct outputs), the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

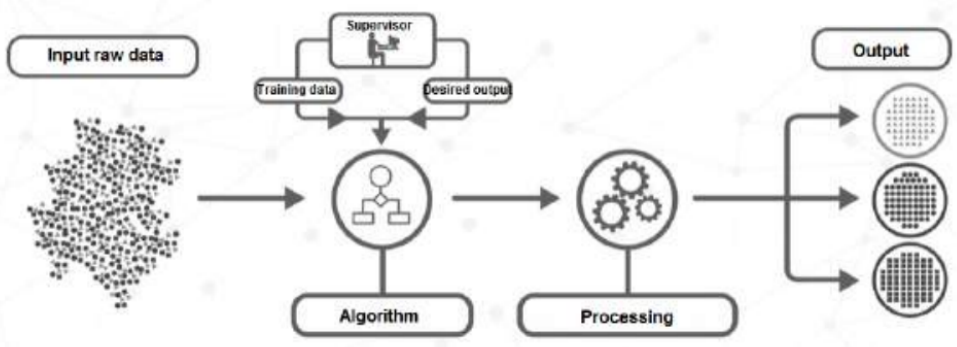


Figure 2: Fig 2:Supervised learning

Example:
Consider the following data regarding patients entering a clinic. The data consists of the gender and age of the patients and each patient is labeled as "healthy" or "sick".

Based on this data, when a new patient enters the clinic, how can one predict whether he/she is healthy or sick?

2. **Unsupervised learning**

gender	age	label
M	48	sick
M	67	sick
F	53	healthy
M	49	healthy
F	34	sick
M	21	healthy

- Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses.
- In unsupervised learning algorithms, a classification or categorization is not included in the observations.
- There are no output values and so there is no estimation of functions. Since the examples given to the learner are unlabeled, the accuracy of the structure that is output by the algorithm cannot be evaluated.
- The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data

Example Consider the following data regarding patients entering a clinic. The data consists of the gender and age of the patients.

gender	age
M	48
M	67
F	53
M	49
F	34
M	21

Based on this data, can we infer anything regarding the patients entering the clinic?

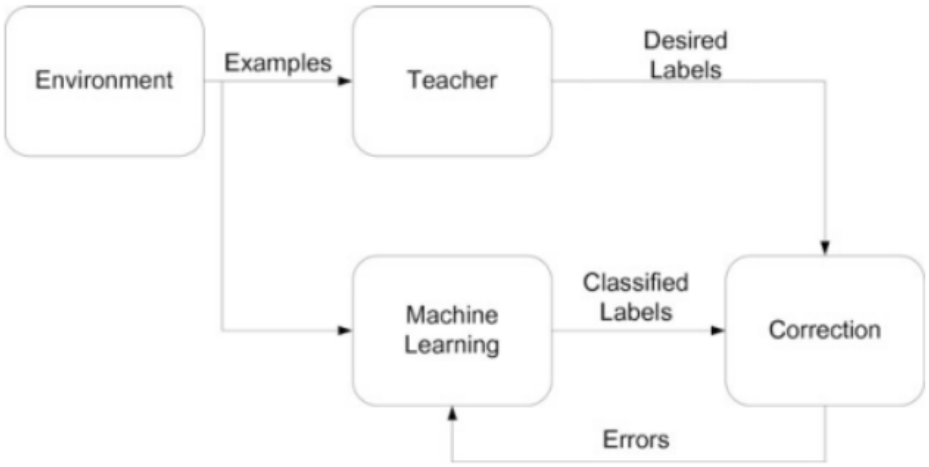
3. **Reinforcement learning**

- Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards.
- A learner (the program) is not told what actions to take as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them.
- In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situations and, through that, all subsequent rewards.
- **For example:**, consider teaching a dog a new trick: we cannot tell it what to do, but we can reward/punish it if it does the right/wrong thing. It has to find out what it did that made it get the reward/punishment. We can use a similar method to train computers to do many tasks, such as playing backgammon or chess, scheduling jobs, and controlling robot limbs.

- Reinforcement learning is different from supervised learning. Supervised learning is learning from examples provided by a knowledgeable expert.

2 Supervised Learning

In supervised learning, the training data you feed to the algorithm includes the desired solutions, called labels. These methods use a training set that consists of labeled data points (for which we know the correct label values). We refer to a data point as labeled if its label value is known. Labeled data points might be obtained from human experts that annotate ("label") data points with their label values.



- Supervised learning is a machine learning paradigm where algorithms aim to optimize parameters to minimize the difference between target and computed outputs, commonly used in tasks like classification and regression.
- In supervised learning, training examples are associated with target outputs (initially labeled) and computed outputs (generated by the learning algorithm), and the goal is to minimize misclassification or error.
- The learning process in supervised learning involves initializing parameters randomly, computing output values for labeled examples, and updating parameters iteratively to minimize errors or achieve convergence.

Type of Supervised Learning:
1.Regression
2.Classification

3 Regression:

- In machine learning, a regression problem is the problem of predicting the value of a numeric variable based on observed values of the variable.
- The value of the output variable may be a number, such as an integer or a floating point value. These are often quantities, such as amounts and sizes. The input variables may be discrete or real-valued.
- Regression algorithms are used if there is a relationship between the input variable and the output variable.
- It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc.

General Approach
Let x denote the set of input variables and y the output variable. In machine learning, the general approach to regression is to assume a model, that is, some mathematical relation between x and y, involving some parameters say, θ, in the following form:
$y = f(x, \theta)$
The function $f(x, \theta)$ is called the regression function. The machine learning algorithm optimizes the parameters in the set θ such that the approximation error is minimized; that is, the estimates of the values of the dependent variable y are as close as possible to the correct values given in the training set.

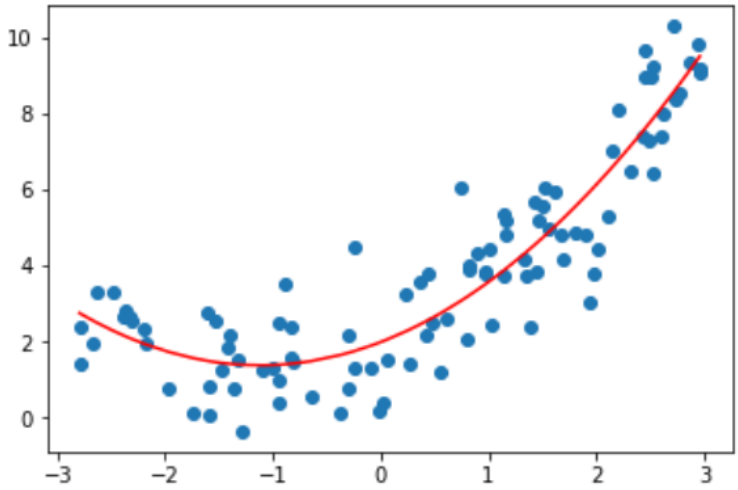
Example
For example, if the input variables are "Age", "Distance" and "Weight" and the output variable is "Price", the model may be

$$y = f(x, \theta)$$
$$\text{Price} = a_0 + a_1 \times (\text{Age}) + a_2 \times (\text{Distance}) + a_3 \times (\text{Weight})$$

where x =(Age, Distance, Weight) denotes the set of input variables and $\theta = (a_0, a_1, a_2, a_3)$ denotes the set of parameters of the model.
Various types of regression techniques These techniques mostly differ in three aspects, namely, the number and type of independent variables, the type of dependent variables and the shape of regression line. Some of these are listed below.

1. Simple linear regression: There is only one continuous independent variable x and the assumed relation between the independent variable and the dependent variable y is
   $y = a + bx$
2. Multivariate linear regression: There are more than one independent variable, say $x1, ..., xn$, and the assumed relation between the independent variables and the dependent variable is
   $y = a_0 + a_1x_1 + ... + a_nx_n$

3. Polynomial regression: There is only one continuous independent variable x and the assumed model is $y = a_0 + a_1x + ... + a_nx^n$
   It is a variant of the multiple linear regression model, except that the best fit line is curved rather than straight.



4. Ridge regression: Ridge regression is one of the types of linear regression in which a small amount of bias is introduced so that we can get better long-term predictions.
   Ridge regression is a regularization technique, which is used to reduce the complexity of the model. It is also called as L2 regularization.

5. Logistic regression: The dependent variable is binary, that is, a variable which takes only the values 0 and 1. The assumed model involves certain probability distributions.

3.0.1 Linear Regression

Simple linear regression is a basic machine learning technique used for modeling the relationship between a single independent variable (often denoted as "x") and a dependent variable (often denoted as "y"). It assumes a linear relationship between the variables and aims to find the best-fitting line (typically represented by the equation $y = mx + b$) that minimizes the sum of squared differences between the observed data points and the values predicted by the model.

Equation: The linear regression model is represented by the equation
$y = ax + b$,
where:
y is the dependent variable (the one you want to predict).
x is the independent variable (the one used for prediction).
a is the slope (also called the regression coefficient), representing how much y changes for each unit change in x.
b is the y-intercept, representing the value of y when x is 0.

In order to determine the optimal estimates of a and b, an estimation method known as Ordinary Least Squares (OLS) is used.

Formulas to find a and b

The means of x and y are given by

$$\bar{x} = \frac{1}{n}\sum x_i$$

$$\bar{y} = \frac{1}{n}\sum y_i$$

and also that the variance of x is given by

$$\text{Var}(x) = \frac{1}{n-1}\sum (x_i - \bar{x_i})^2.$$

The covariance of x and y, denoted by Cov(x, y) is defined as

$$\text{Cov}(x, y) = \frac{1}{n-1}\sum (x_i - \bar{x})(y_i - \bar{y})$$

It can be shown that the values of a and b can be computed using the following formulas:

$$b = \frac{\text{Cov}(x, y)}{\text{Var}(x)}$$

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

Advantages of Simple Linear Regression:

- Simplicity and ease of interpretation.
- Transparent modeling with clear coefficient interpretations.
- Computational efficiency, suitable for large datasets.
- A baseline model for assessing feature significance.
- Effective when the relationship between variables is linear.

Disadvantages of Simple Linear Regression:

- Limited to linear relationships, may perform poorly for nonlinear data.
- Sensitive to outliers, leading to parameter influence.
- Prone to underfitting when facing complex relationships.
- Assumptions of independent and normally distributed errors are critical.
- Suitable only when one independent variable is involved in the analysis.

Example

Obtain a linear regression for the data in below table assuming that y is the independent variable.

x	1.0	2.0	3.0	4.0	5.0
y	1.00	2.00	1.30	3.75	2.25

$$
\begin{aligned}
n &= 5 \\
\bar{x} &= \frac{1}{5}(1.0 + 2.0 + 3.0 + 4.0 + 5.0) \\
&= 3.0 \\
\bar{y} &= \frac{1}{5}(1.00 + 2.00 + 1.30 + 3.75 + 2.25) \\
&= 2.06 \\
\text{Cov}\,(x, y) &= \frac{1}{4}[(1.0 - 3.0)(1.00 - 2.06) + \cdots + (5.0 - 3.0)(2.25 - 2.06)] \\
&= 1.0625 \\
\text{Var}\,(x) &= \frac{1}{4}[(1.0 - 3.0)^2 + \cdots + (5.0 - 3.0)^2] \\
&= 2.5 \\
b &= \frac{1.0625}{2.5} \\
&= 0.425 \\
a &= 2.06 - 0.425 \times 3.0 \\
&= 0.785
\end{aligned}
$$

Therefore, the linear regression model for the data is $y = 0.785 + 0.425x$

3.0.2 Multivariate Linear Regression

Multiple Linear Regression is a machine learning technique used to model the relationship between a dependent variable (target) and multiple independent variables (features) by fitting a linear equation to the data.
The model can be expressed as: $y = \beta_0 + \beta_1 x_1 + ... + \beta_N x_N$
Where:
y is the dependent variable (the one you want to predict).
$x_1, x_2, ..., x_n$ are the independent variables.
β_0 is the y-intercept.
$\beta_1, \beta_2, ...\beta_n$ are the coefficients associated with each independent variable.

Let there also be n observed values of these variables:

Variables (features)	Values (examples)			
	Example 1	Example 2	⋯	Example n
x_1	x_{11}	x_{12}	⋯	x_{1n}
x_2	x_{21}	x_{22}	⋯	x_{2n}
⋯				
x_N	x_{N1}	x_{N2}	⋯	x_{Nn}
y (outcomes)	y_1	y_2	⋯	y_n

As in simple linear regression, here also we use the ordinary least squares method to obtain the optimal estimates of $\beta_0, \beta_1, ...\beta_n$ The method yields the following procedure for the computation of these optimal estimates. Let

$$X = \begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{N1} \\ 1 & x_{12} & x_{22} & \cdots & x_{N2} \\ \vdots & & & & \\ 1 & x_{1n} & x_{2n} & \cdots & x_{Nn} \end{bmatrix}, \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad B = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_N \end{bmatrix}$$

Then it can be shown that the regression coefficients are given by
$B = (X^T X)^{-1} X^T Y$

Advantages:

- The multivariate regression method helps you find a relationship between multiple variables or features.
- It also defines the correlation between independent variables and dependent variables.

Disadvantages:

- Multivariate regression technique requires high-level mathematical calculations.
- It is complex.
- The output of the multivariate regression model is difficult to analyse.
- The loss can use errors in the output.
- Multivariate regression yields better results when used with larger datasets rather than small ones.

Example:
Fit a multiple linear regression model to the following data:

x_1	1	1	2	0
x_2	1	2	2	1
y	3.25	6.5	3.5	5.0

In this problem, there are two independent variables and four sets of values of the variables. Thus, in the notations used above, we have n = 2 and N = 4. The multiple linear regression model for this problem has the form $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

$$X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 2 \\ 1 & 0 & 1 \end{bmatrix}, \quad Y = \begin{bmatrix} 3.25 \\ 6.5 \\ 3.5 \\ 5.0 \end{bmatrix}, \quad B = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 4 & 4 & 6 \\ 4 & 6 & 7 \\ 6 & 7 & 10 \end{bmatrix}$$

$$(X^T X)^{-1} = \begin{bmatrix} \frac{11}{4} & \frac{1}{2} & -2 \\ \frac{1}{2} & 1 & -1 \\ -2 & -1 & 2 \end{bmatrix}$$

$$B = (X^T X)^{-1} X^T Y$$

$$= \begin{bmatrix} 2.0625 \\ -2.3750 \\ 3.2500 \end{bmatrix}$$

The required model is $y = 2.0625 - 2.3750x_1 + 3.2500x_2$

3.0.3 Ridge Regression

(Important for GATE DA)

Ridge Regression, also known as L2 regularization, is a machine learning technique used to mitigate the issues of multicollinearity and overfitting in Multiple Linear Regression. It adds a regularization term to the regression equation, modifying the loss function, and aims to minimize the sum of squared errors along with the sum of the squared coefficients, which can be expressed as:

$$\sum_{i=1}^{M}(y_i - y'_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{n}\beta_j * x_{ij}\right)^2 + \lambda\sum_{j=0}^{n}\beta_j{}^2$$

- When data exhibits multicollinearity, that is, the ridge regression technique is applied when the independent variables are highly correlated. While least squares estimates are unbiased in multicollinearity, their variances are significant enough to cause the observed value to diverge from the actual value. Ridge regression reduces standard errors by biassing the regression estimates.
- The lambda (λ) variable in the ridge regression equation resolves the multicollinearity problem.
- Lambda (λ) is the penalty term. So, by changing the values of (λ), we are controlling the penalty term. The higher the values of (λ), the bigger is the penalty and therefore the magnitude of coefficients is reduced.
- In this technique, the cost function is altered by adding the penalty term to it. The amount of bias added to the model is called **Ridge Regression penalty**. We can calculate it by multiplying with the lambda to the squared weight of each individual feature.
- In the above equation, the penalty term regularizes the coefficients of the model, and hence ridge regression reduces the amplitudes of the coefficients that decreases the complexity of the model.
- As we can see from the above equation, if the values of (λ) tend to zero, the equation becomes the cost function of the linear regression model. Hence, for the minimum value of (λ) , the model will resemble the linear regression model.
- A general linear or polynomial regression will fail if there is high collinearity between the independent variables, so to solve such problems, Ridge regression can be used.
- It helps to solve the problems if we have more parameters than samples.

Bias and variance trade-off

Bias and variance trade-off is generally complicated when it comes to building ridge regression models on an actual dataset. However, following the general trend which one needs to remember is:

- The bias increases as (λ) increases.

- The variance decreases as (λ) increases.

Assumptions of Ridge Regressions:
The assumptions of ridge regression are the same as that of linear regression: linearity, constant variance, and independence. However, as ridge regression does not provide confidence limits, the distribution of errors to be normal need not be assumed.

- **Linear Relationship:** Ridge Regression assumes that there is a linear relationship between the independent variables and the dependent variable.
- **Homoscedasticity:** Ridge Regression assumes that the variance of the errors is constant across all levels of the independent variables.
- **Independence of errors:** Ridge Regression assumes that the errors are independent of each other, i.e., the errors are not correlated.
- **Normality of errors:** Ridge Regression assumes that the errors follow a normal distribution.

Key points about Ridge Regression in machine learning:

- **Regularization:** Ridge regression adds a penalty term that discourages the magnitude of the coefficients from becoming too large. This helps prevent overfitting.
- **Multicollinearity Mitigation:** It's particularly effective when you have highly correlated independent variables (multicollinearity) by shrinking the coefficients and making the model more stable.
- **Lambda Parameter:** The choice of the lambda parameter (λ) is essential. A small λ is close to standard linear regression, while a large λ results in stronger regularization.
- **Balancing Act:** Ridge regression performs a balancing act between fitting the data well and preventing overfitting. It maintains all predictors but assigns smaller coefficients to less important ones.
- **Model Stability:** It makes the model more stable, especially when you have a high-dimensional dataset with many predictors. This can lead to better generalization to new, unseen data.
- **Interpretability:** Ridge regression may make the model less interpretable because it shrinks coefficients toward zero. It can be challenging to discern the individual importance of predictors.
- **Tuning Lambda:** Cross-validation is often used to tune the lambda parameter and find the optimal trade-off between fitting the data and regularization.

Disadvantages of Ridge Regression:

- **Sensitivity to Lambda:** Proper selection of the regularization parameter (λ) is crucial; an incorrect choice can result in underfitting or ineffective regularization.

- **Loss of Interpretability:** Ridge regression can make the model less interpretable since it shrinks coefficients towards zero, potentially making it harder to discern the individual predictor's importance.
- **Ineffective for Feature Selection:** Ridge regression does not perform feature selection. It retains all predictors in the model but assigns smaller coefficients to less important ones.
- **Less Effective for Sparse Data:** In cases where many predictors are irrelevant or unimportant, Ridge may not eliminate them from the model effectively.

Extra info to understand more about Ridge Regression

What is Regularization?

- Regularization is one of the most important concepts of machine learning. It is a technique to prevent the model from overfitting by adding extra information to it.
- Sometimes the machine learning model performs well with the training data but does not perform well with the test data. It means the model is not able to predict the output when deals with unseen data by introducing noise in the output, and hence the model is called overfitted. This problem can be deal with the help of a regularization technique.
- This technique can be used in such a way that it will allow to maintain all variables or features in the model by reducing the magnitude of the variables. Hence, it maintains accuracy as well as a generalization of the model.
- It mainly regularizes or reduces the coefficient of features toward zero. In simple words, "In regularization technique, we reduce the magnitude of the features by keeping the same number of features."

What is Shrinkage?

- Shrinkage refers to the process of shrinking the estimated regression coefficients towards zero. This is done by adding a penalty term to the sum of squared residuals in the regression equation, which is called the regularization term.
- The regularization term is proportional to the square of the magnitude of the regression coefficients, and it is controlled by a tuning parameter, usually denoted as λ. The higher the value of λ, the more the coefficients are shrunk towards zero.
- Shrinkage helps to reduce the variance of the estimates and can improve the prediction accuracy of the model.

What is Multicollinearity?

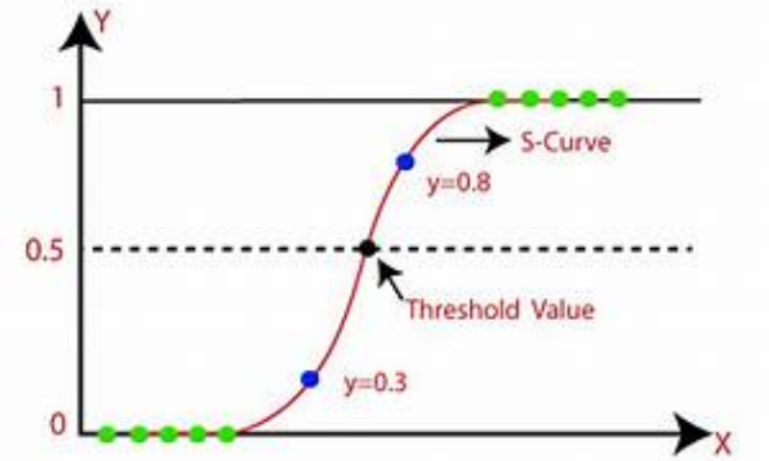
- Multicollinearity is a phenomenon where one predicted value in several regression models is linearly predicted with others.
- Multicollinearity basically happens when more than two anticipated variables have substantial correlations with one another.
- In modeled data, multicollinearity could be defined as the presence of a correlation between independent variables. Estimates of the regression coefficient may become inaccurate as a result. It can potentially raise the standard errors of the regression coefficients and reduce the efficacy of any t-tests.
- In addition to increasing model redundancy and decreasing predictability's effectiveness and dependability, multicollinearity can provide false results and p-values.
- Multicollinearity can be introduced by using multiple data sources. This could happen as a result of limitations placed on linear or demographic models, an overly precise model, outliers, or model design or choice made during the data collection process.
- Multicollinearity may be introduced during the data collection process if the data were gathered using an inappropriate sampling method. Even if the sample size is smaller than expected, it could still happen.
- Because there are more variables than data, multicollinearity will be visible if the model is overspecified.

4 Logistic Regression

- Logistic Regression is a machine learning algorithm used for binary classification tasks, modeling the probability of an event occurring or not, by fitting a logistic curve to the data. It's expressed as the logistic function, which maps the linear combination of input features to values between 0 and 1.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1). The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification.
- It is used for predicting the categorical dependent variable using a given set of independent variables.

4.1 Logistic Function (Sigmoid Function)

$$y = \frac{1}{1 + e^{-z}}$$



- The sigmoid function is a mathematical function used to map the predicted values to probabilities. It maps any real value into another value within a range of 0 and 1. o The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form.
- The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

4.2 Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

- **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
- **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

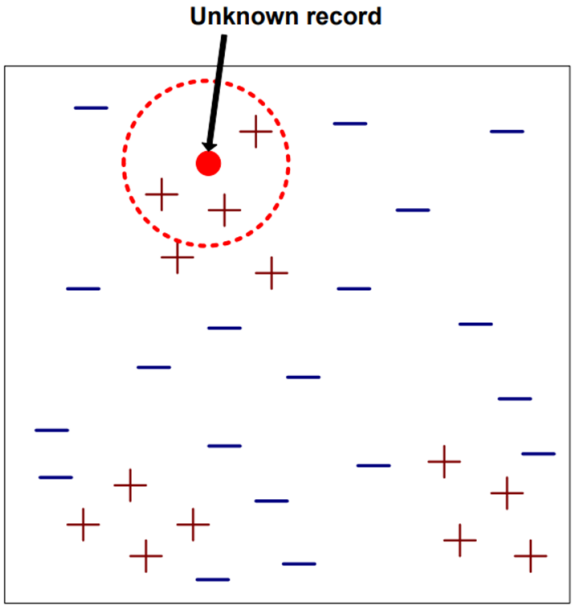
Key points about Logistic Regression include:

- **Binary Classification:** It's primarily used for two-class classification problems, where the output is either 0 or 1, indicating the absence or presence of an event.
- **Logistic Function:** Utilizes the logistic (sigmoid) function to convert a linear combination of input features into a probability value between 0 and 1.
- **Coefficient Interpretation:** Coefficients represent the impact of each feature on the probability of the event, making the model interpretable.
- **Maximum Likelihood Estimation:** The model is trained using maximum likelihood estimation to find the optimal parameters that best fit the data.
- **Decision Boundary:** Logistic Regression calculates a decision boundary to separate the classes, making it a linear classifier by default.
- **Regularization:** Regularization techniques, such as L1 (Lasso) and L2 (Ridge), can be applied to prevent overfitting and improve model generalization.
- **Evaluating Performance:** Common performance metrics include accuracy, precision, recall, F1 score, and the ROC-AUC curve.
- **Extensions:** Multinomial (softmax) logistic regression is used for multi-class classification tasks.
- **Applications:** Logistic Regression is applied in various fields, including medical diagnosis, finance, spam detection, and sentiment analysis.

5 K-Nearest Neighbors

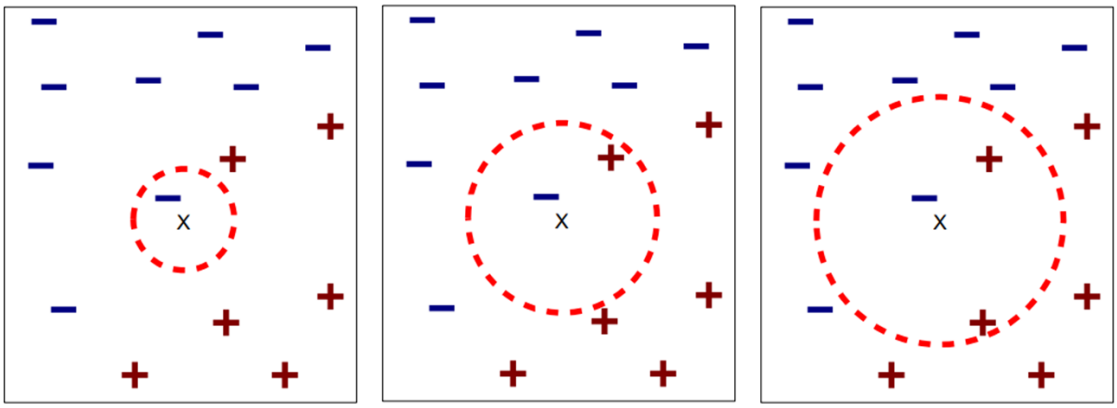
- K-Nearest Neighbors (KNN) is a simple and intuitive machine-learning algorithm used for both classification and regression tasks.
- It is a non-parametric and instance-based learning method, which means it doesn't make any assumptions about the underlying data distribution and makes predictions based on the similarity of data points.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points which is closest to the test data. The KNN algorithm calculates the probability of the test data belonging to the classes of 'K' training data and class that holds the highest probability will be selected. In the case of regression, the value is the mean of the 'K' selected training points.

Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor

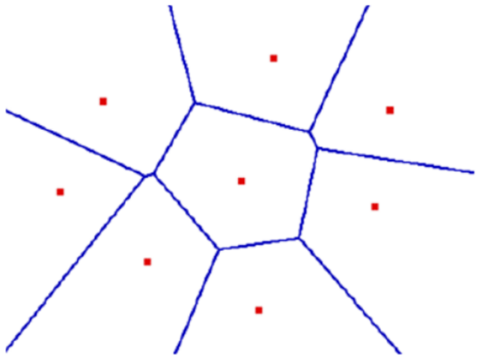


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

Voronoi diagram, deals with when value of K =1

Describes the areas that are nearest to any given point, given a set of data. Each line segment is equidistant between two points of opposite class



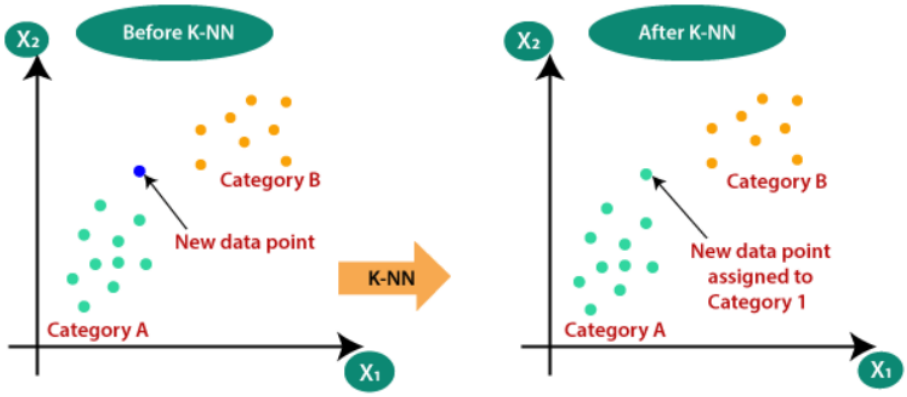
Choosing Value of K

- Larger k may lead to better performance But if we set k too large we may end up looking at samples that are not neighbors (are far away from the query)
- We can use cross-validation to find k
- Rule of thumb is k ¡ sqrt(n), where n is the number of training example
- Larger k produces smoother boundary effect
- When K==N, always predict the majority class

5.1 Working

The K-NN working can be explained on the basis of the below algorithm:

1. Select the number K of the neighbors
2. Calculate the Euclidean distance of K number of neighbors
3. Take the K nearest neighbors as per the calculated Euclidean distance.
4. Among these k neighbors, count the number of the data points in each category.
5. Assign the new data points to that category for which the number of the neighbor is maximum.
6. Our model is ready.



5.2 K-Nearest Neighbors (KNN) Classification Example

Suppose we have a dataset with the following points:

Data Point	Feature 1 (X1)	Feature 2 (X2)	Class
A	1	2	Blue
B	2	3	Blue
C	2	1	Red
D	3	3	Red
E	4	2	Blue

Now, let's say we want to classify a new data point with features $X1 = 2.5$ and $X2 = 2.5$ using a KNN algorithm with $k = 3$ (i.e., considering the three nearest neighbors).

1. Calculate Euclidean Distances:

$$\text{Distance to A: } \sqrt{(2.5-1)^2 + (2.5-2)^2} = \sqrt{1.5}$$
$$\text{Distance to B: } \sqrt{(2.5-2)^2 + (2.5-3)^2} = \sqrt{1.5}$$
$$\text{Distance to C: } \sqrt{(2.5-2)^2 + (2.5-1)^2} = \sqrt{1.5}$$
$$\text{Distance to D: } \sqrt{(2.5-3)^2 + (2.5-3)^2} = \sqrt{2.5}$$
$$\text{Distance to E: } \sqrt{(2.5-4)^2 + (2.5-2)^2} = \sqrt{4.5}$$

2. Find K Nearest Neighbors: Identify the three nearest neighbors based on the calculated distances. In this case, the three closest points are A, B, and C.

3. Majority Voting: Determine the majority class among the three nearest neighbors. Since A and B are Blue, and C is Red, the majority class is Blue.

4. Prediction: Predict that the new point $X1 = 2.5, X2 = 2.5$ belongs to the majority class, which is Blue.

5.3 Advantages of K-Nearest Neighbors (KNN):

- **Simplicity:** KNN is easy to understand and implement, making it a good choice for simple classification and regression tasks.
- **No Training Period:** KNN is a lazy learner, meaning it doesn't require a lengthy training period. It stores the entire training dataset and makes predictions when needed.

- **Non-parametric:** KNN doesn't make any assumptions about the underlying data distribution, making it versatile for a wide range of applications.
- **Adaptability:** KNN can be used for both classification and regression tasks, and it can handle multi-class problems without modification.
- **Interpretability:** The algorithm provides human-interpretable results, as predictions are based on the majority class or the average of the nearest neighbors.

5.4 Disadvantages of K-Nearest Neighbors (KNN):

- **Computational Intensity:** KNN can be computationally expensive, especially for large datasets, as it requires calculating distances to all data points during prediction.
- **Sensitivity to Feature Scaling:** KNN is sensitive to the scale of features, so it's essential to normalize or standardize your data before applying the algorithm.
- **Curse of Dimensionality:** KNN's performance degrades as the number of features or dimensions increases, as distances between data points become less meaningful in high-dimensional spaces.
- **Determining the Optimal K:** Selecting the right value for K is crucial, and choosing an inappropriate K can lead to underfitting or overfitting. There's no universally optimal value, and it often requires experimentation.
- **Imbalanced Data:** KNN can be biased towards the majority class in imbalanced datasets. It's essential to balance the dataset or adjust the class weights when necessary.

6 Naive Bayes Classifier

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

6.1 Bayes' Theorem

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability. The formula for Bayes' theorem is given as:

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

Where,

P(A—B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B—A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true. P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Assumption

The fundamental Naive Bayes assumption is that each feature makes an **independent** and **equal** contribution to the outcome.

With relation to our dataset, this concept can be understood as:

We assume that no pair of features are dependent. For example, the temperature being 'Hot' has nothing to do with the humidity or the outlook being 'Rainy' has no effect on the winds. Hence, the features are assumed to be independent.

Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can't predict the outcome accurately. None of the attributes is irrelevant and assumed to be contributing equally to the outcome.

Here are the key concepts and characteristics of the Naive Bayes classifier:

- **Bayes' Theorem:** The classifier is based on Bayes' theorem, which calculates the probability of a hypothesis (in this case, a class label) given the evidence (features or attributes). Mathematically, it is expressed as P(class—evidence) = [P(evidence—class) * P(class)] / P(evidence).
- **Independence Assumption:** The "Naive" in Naive Bayes refers to the assumption that all features are independent of each other, given the class label. In reality, this assumption

is often not true, but the simplification makes the algorithm computationally efficient and easy to implement.

- **Types of Naive Bayes Classifiers:**
  1. Multinomial Naive Bayes: Typically used for text classification where features represent word counts.
  2. Gaussian Naive Bayes: Suitable for continuous data and assumes a Gaussian distribution of features.
  3. Bernoulli Naive Bayes: Applicable when features are binary, such as presence or absence.
- **Prior Probability (P(class)):** This is the probability of a class occurring before observing any evidence. It can be calculated from the training data.
- **Likelihood (P(evidence—class)):** This represents the probability of observing a specific set of features given a class label. For text classification, this might involve counting the occurrence of words in documents.
- **Posterior Probability (P(class—evidence)):** It is the probability of a class label given the evidence, which is what the Naive Bayes classifier calculates for each class.
- **Classification:** To classify a new data point, the classifier calculates the posterior probabilities for each class and selects the class with the highest probability.

Bayes Classifiers

- Bayesian classifiers use **Bayes theorem**, which says

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

- $p(c_j \mid d)$ = probability of instance d being in class c_j,
  This is what we are trying to compute
- $p(d \mid c_j)$ = probability of generating instance d given class c_j,
  We can imagine that being in class c_j, causes you to have feature d with some probability
- $p(c_j)$ = probability of occurrence of class c_j,
  This is just how frequent the class c_j, is in our database
- $p(d)$ = probability of instance d occurring

  This can actually be ignored, since it is the same for all classes

  - To simplify the task, **naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

$$p(d|c_j) = p(d_1|c_j) * p(d_2|c_j) * \ldots * p(d_n|c_j)$$

The probability of class c_j generating instance d, equals….

The probability of class c_j generating the observed value for feature 1, multiplied by..

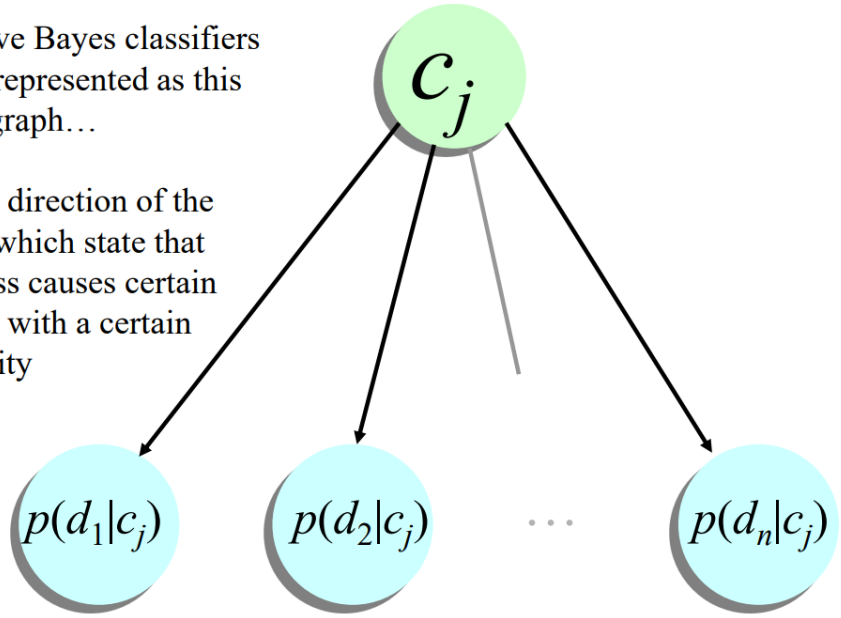
The probability of class c_j generating the observed value for feature 2, multiplied by..

- To simplify the task, **naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

$$p(d|c_j) = p(d_1|c_j) * p(d_2|c_j) * \ldots * p(d_n|c_j)$$

The Naive Bayes classifiers is often represented as this type of graph…

Note the direction of the arrows, which state that each class causes certain features, with a certain probability

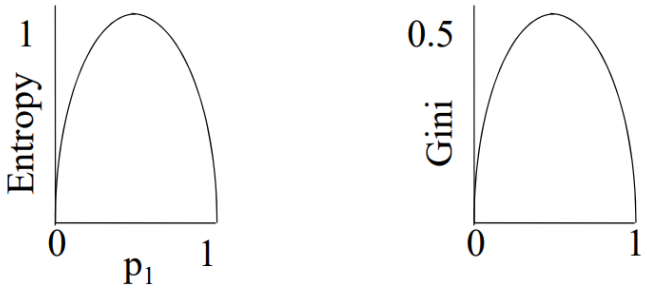


- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes theorem

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

  - Choose value of C that maximizes
    $P(C \mid A_1, A_2, \ldots, A_n)$
  - Equivalent to choosing value of C that maximizes
    $P(A_1, A_2, \ldots, A_n|C)\ P(C)$

- How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?



- Information gain on partitioning S into r subsets
- Impurity (S) - sum of weighted impurity of each subset

$$Gain(S, S_1..S_r) = Entropy(S) - \sum_{j=1}^{r} \frac{S_j}{S} Entropy(S_j)$$

6.2 Advantages of Naive Bayes:

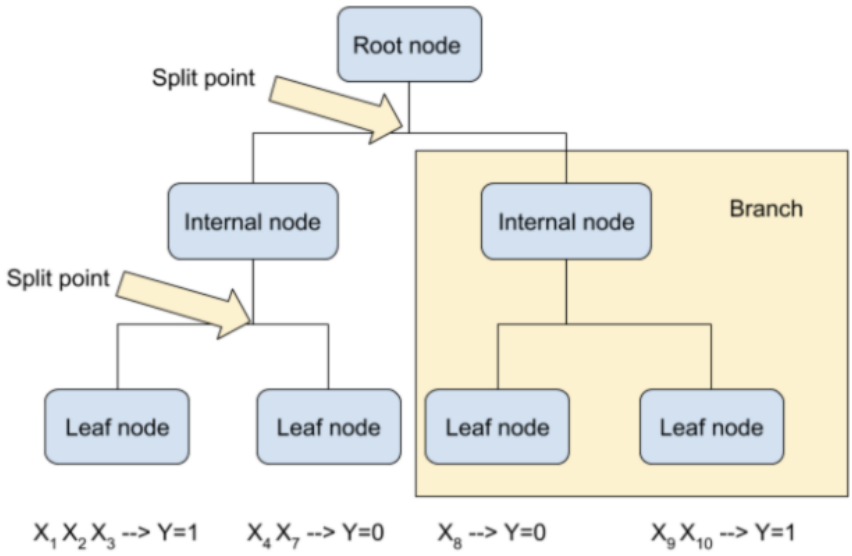
- Simplicity: Naive Bayes is easy to implement and understand, making it a good choice for quick classification tasks.
- Efficiency: It can handle a large number of features efficiently, particularly in text classification.
- Works Well with Small Datasets: It can perform reasonably well even with limited training data.
- Multiclass Classification: It can be used for multiclass classification problems.
- Interpretable: The results are easy to interpret, as it provide the probability of belonging to each class.

6.3 Disadvantages of Naive Bayes:

- **Independence Assumption:** The assumption of feature independence doesn't always hold, which can affect accuracy.
- Sensitivity to Feature Distribution: It may not perform well when features have complex, non-Gaussian distributions.
- Requires Sufficient Data: For some cases, Naive Bayes might not perform well when there is a scarcity of data.
- Zero Probability Problem: If a feature-class combination does not exist in the training data, the probability will be zero, causing issues. Smoothing techniques are often used to address this.

7 Decision Trees

- A decision tree is a simple model for supervised classification. It is used for classifying a single discrete target feature.
- Each internal node performs a Boolean test on an input feature (in general, a test may have more than two options, but these can be converted to a series of Boolean tests). The edges are labeled with the values of that input feature.
- Each leaf node specifies a value for the target feature
- Classifying an example using a decision tree is very intuitive. We traverse down the tree, evaluating each test and following the corresponding edge. When a leaf is reached, we return the classification on that leaf.
- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.
- It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.



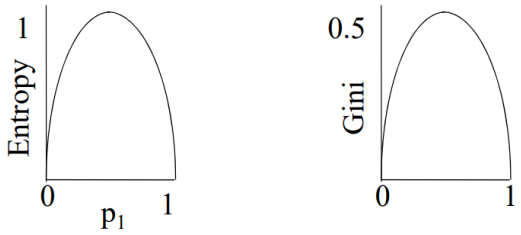
7.1 Terminologies

- **Root Node:** A decision tree's root node, which represents the original choice or feature from which the tree branches, is the highest node.
- **Internal Nodes (Decision Nodes):** Nodes in the tree whose choices are determined by the values of particular attributes. There are branches on these nodes that go to other nodes.
- **Leaf Nodes (Terminal Nodes):** The branches' termini, when choices or forecasts are decided upon. There are no more branches on leaf nodes.
- **Branches (Edges):** Links between nodes that show how decisions are made in response to particular circumstances.
- **Splitting:** The process of dividing a node into two or more sub-nodes based on a decision criterion. It involves selecting a feature and a threshold to create subsets of data.
- **Parent Node:** A node that is split into child nodes. The original node from which a split originates.
- **Child Node:** Nodes created as a result of a split from a parent node.
- **Decision Criterion:** The rule or condition used to determine how the data should be split at a decision node. It involves comparing feature values against a threshold.
- **Pruning:** The process of removing branches or nodes from a decision tree to improve its generalization and prevent overfitting.

7.2 Measures of impurity

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

1. Information Gain



- Information gain on partitioning S into r subsets
- Impurity (S) - sum of weighted impurity of each subset

$$Gain(S, S_1..S_r) = Entropy(S) - \sum_{j=1}^{r} \frac{S_j}{S} Entropy(S_j)$$

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first.

It can be calculated using the below formula:
Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)
Entropy: Entropy is a metric to measure the impurity in a given attribute.

$$Entropy \quad (S) = -\sum_{i=1}^{k} p_i \log \; p_i$$

It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

2. Gini Index

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with a low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.

Gini index can be calculated using the below formula:

$$Gini\ (S) = 1 - \sum_{i=1}^{k} p_i^{\,2}$$

7.3 Advantages of Decision Trees:

- Interpretability: Decision Trees are easy to interpret, making them a good choice when you need to explain or visualize the model's decisions.
- Handling Non-linearity: Decision Trees can capture non-linear relationships between features and the target variable.
- Feature Selection: They can automatically select the most important features, reducing the need for feature engineering.
- Versatility: Decision Trees can handle both categorical and numerical data.
- Efficiency: They are relatively efficient during prediction, with time complexity logarithmic in the number of data points.

7.4 Disadvantages of Decision Trees:

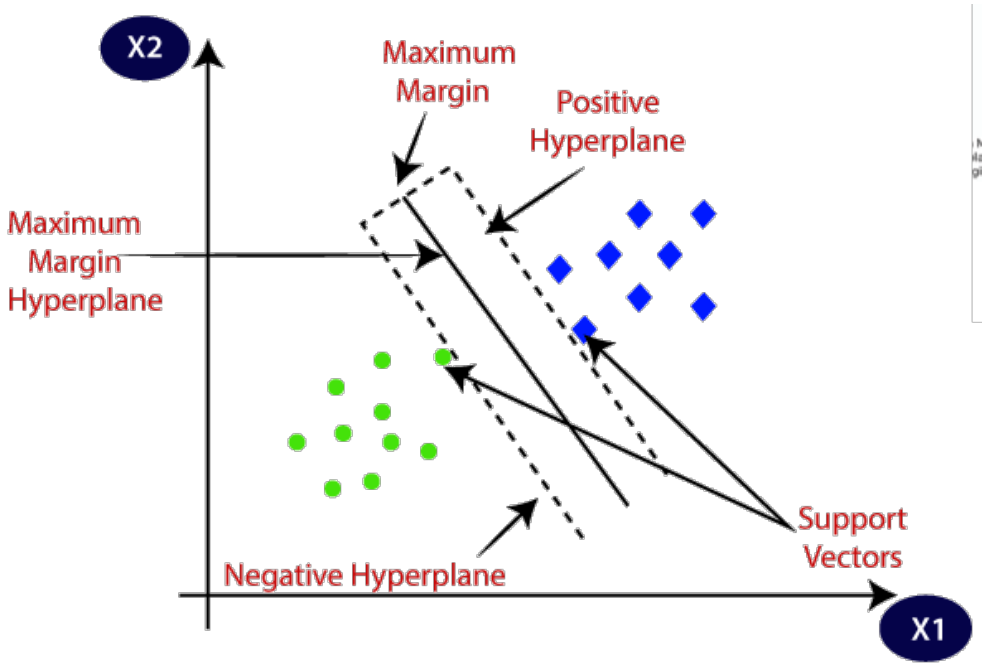
- Overfitting: Decision Trees can be prone to overfitting, creating complex models that don't generalize well to new data. Pruning and setting appropriate parameters can help mitigate this.
- Bias Toward Dominant Classes: In classification tasks, Decision Trees can be biased toward dominant classes, leading to imbalanced predictions.
- Instability: Small variations in the data can lead to different tree structures, making them unstable models.
- Greedy Algorithm: Decision Trees use a greedy algorithm, making locally optimal decisions at each node, which may not lead to the global optimal tree structure

8 Support Vector Machine

- Support Vector Machine is a system for efficiently training linear learning machines in kernel-induced feature spaces, while respecting the insights of generalisation theory and exploiting optimisation theory.'
- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.
- SVMs pick best separating hyperplane according to some criterion e.g. maximum margin

- Training process is an optimisation
- Training set is effectively reduced to a relatively small number of support vectors
- SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

  Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Here are the key concepts and characteristics of Support Vector Machines:

- **Hyperplane:** There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.
- In a binary classification problem, an SVM finds a hyperplane that best separates the data points of different classes. This hyperplane is the decision boundary.
- The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

  We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.
- **Support Vectors:**The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector. They are critical for defining the margin and determining the location of the hyperplane.

- **Margin:** The margin is the distance between the support vectors and the decision boundary. SVM aims to maximize this margin because a larger margin often leads to better generalization.
- **C Parameter:** The regularization parameter "C" controls the trade-off between maximizing the margin and minimizing the classification error. A smaller "C" value results in a larger margin but may allow some misclassifications, while a larger "C" value allows for fewer misclassifications but a smaller margin.
- **Multi-Class Classification:** SVMs are inherently binary classifiers, but they can be extended to handle multi-class classification using techniques like one-vs-one (OvO) or one-vs-all (OvA) classification.
- **The Scalar Product:**The scalar or dot product is, in some sense, a measure of Similarity $a.b = |a|.|b|cos(\theta)$
- **Decision Function for binary classification**

$$f(x) \in \mathbf{R}$$

$$f(x_i) \geq 0 \Rightarrow y_i = 1$$
$$f(x_i) < 0 \Rightarrow y_i = -1$$

- **Feature Spaces** We may separate data by mapping to a higherdimensional feature space – The feature space may even have an infinite number of dimensions!
  • We need not explicitly construct the new feature space

8.1 Kernels

We may use Kernel functions to implicitly map to a new feature space
• Kernel fn: $K(x_1, x_2) \in R$
• Kernel must be equivalent to an inner product in some feature space

- **Kernel Trick:** SVM can handle non-linearly separable data by using a kernel function to map the data into a higher-dimensional space where it becomes linearly separable. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid kernels.

- The linear classifier relies on inner product between vectors $K(\mathbf{x_i},\mathbf{x_j})=\mathbf{x_i}^T\mathbf{x_j}$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x_i},\mathbf{x_j})= \varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j})$$

- A *kernel function* is some function that corresponds to an inner product into some feature space.
- Example:

  2-dimensional vectors $\mathbf{x}=[x_1\ x_2]$; let $K(\mathbf{x_i},\mathbf{x_j})=(1+\mathbf{x_i}^T\mathbf{x_j})^2$,

  Need to show that $K(\mathbf{x_i},\mathbf{x_j})= \varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j})$:

$$K(\mathbf{x_i},\mathbf{x_j})=(1+\mathbf{x_i}^T\mathbf{x_j})^2 = 1+x_{i1}^2x_{j1}^2+2x_{i1}x_{j1}x_{i2}x_{j2}+x_{i2}^2x_{j2}^2+2x_{i1}x_{j1}+2x_{i2}x_{j2}=$$
$$=[1\ \ x_{i1}^2\ \sqrt{2}x_{i1}x_{i2}\ \ x_{i2}^2\ \sqrt{2}x_{i1}\ \sqrt{2}x_{i2}]^T[1\ \ x_{j1}^2\ \sqrt{2}x_{j1}x_{j2}\ \ x_{j2}^2\ \sqrt{2}x_{j1}\ \sqrt{2}x_{j2}]=$$
$$=\varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j}),\quad \text{where } \varphi(\mathbf{x})=[1\ \ x_1^2\ \sqrt{2}x_1x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \sqrt{2}x_2]$$

- For some functions $K(\mathbf{x_i},\mathbf{x_j})$ checking that $K(\mathbf{x_i},\mathbf{x_j})= \varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j})$ can be cumbersome.
- Mercer's theorem:

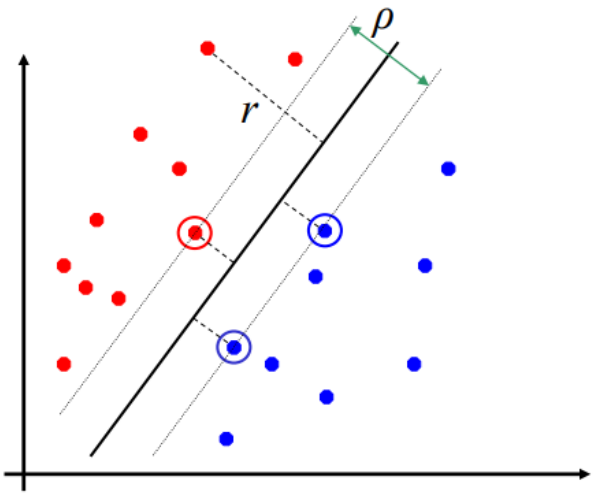
  Every semi-positive definite symmetric function is a kernel

Examples of kernel functions

- Linear: $K(\mathbf{x_i},\mathbf{x_j})= \mathbf{x_i}^T\mathbf{x_j}$
- Polynomial of power p: $K(\mathbf{x_i},\mathbf{x_j})= (1+\mathbf{x_i}^T\mathbf{x_j})^p$
- Gaussian (radial-basis function network): $K(\mathbf{x_i},\mathbf{x_j})= e^{-\frac{\|\mathbf{x_i}-\mathbf{x_j}\|^2}{2\sigma^2}}$
- Two-layer perceptron: $K(\mathbf{x_i},\mathbf{x_j})= \tanh(\beta_0\mathbf{x_i}^T\mathbf{x_j}+\beta_1)$

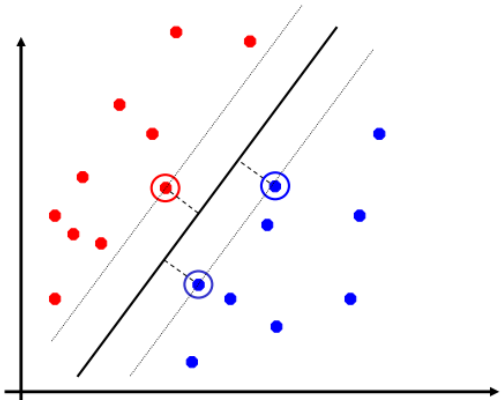
8.2 Classification Margin

- Distance from example data to the separator is $r = \frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$
- Data closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the width of separation between classes.



Maximum Margin Classification

- Maximizing the margin is good according to intuition and theory.
- Implies that only support vectors are important; other training examples are ignorable.

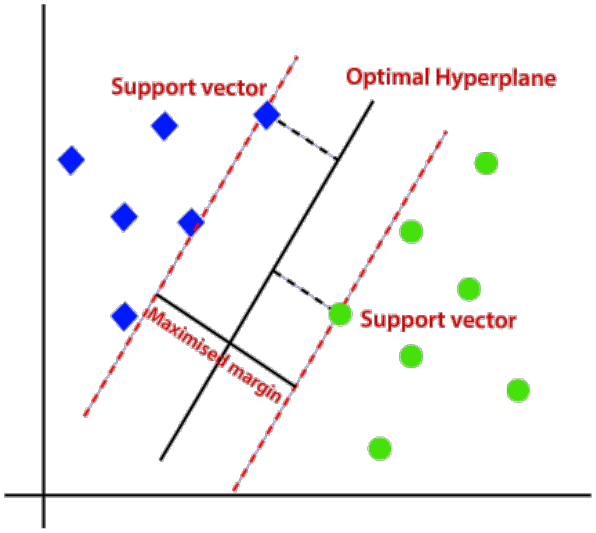


- Misclassification error and the function complexity bound generalization error.
- Maximizing margins minimizes complexity.
- "Eliminates" overfitting.
- Solution depends only on *Support Vectors* not number of attributes.

8.3 Types of SVM

1. Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

   The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes.
   Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a hyperplane. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as margin. And the goal of SVM is to maximize this margin. The hyperplane with the maximum margin is called the optimal hyperplane.



Linear SVM Mathematically

- Assuming all data is at distance larger than 1 from the hyperplane, the following two constraints follow for a training set $\{(\mathbf{x_i}, y_i)\}$

$$\mathbf{w^T x_i} + b \geq 1 \quad \text{if } y_i = 1$$
$$\mathbf{w^T x_i} + b \leq -1 \quad \text{if } y_i = -1$$

- For support vectors, the inequality becomes an equality; then, since each example's distance from the
- hyperplane is $r = \frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$ the margin is: $\rho = \frac{2}{\|\mathbf{w}\|}$

Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem:*

Find **w** and b such that

$\rho = \frac{2}{\|\mathbf{w}\|}$ is maximized and for all $\{(\mathbf{x_i}, y_i)\}$

$\mathbf{w^T x_i} + b \geq 1$ if y_i=1; $\mathbf{w^T x_i} + b \leq -1$ if $y_i = -1$

A better formulation:

Find **w** and b such that

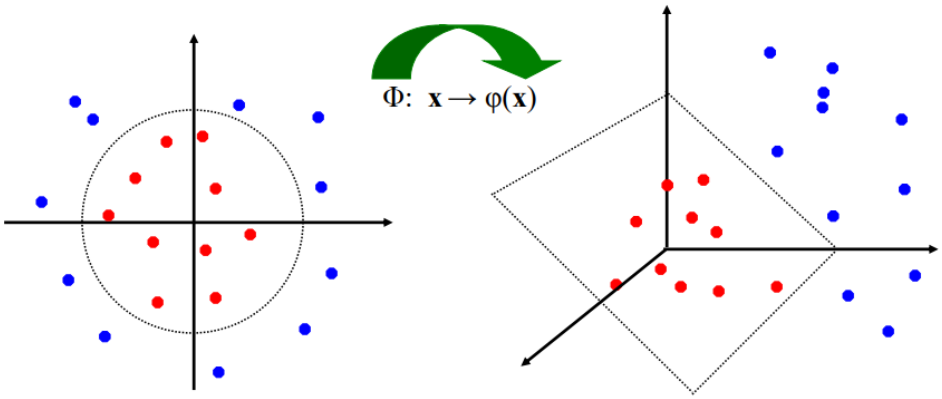
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\,(\mathbf{w^T x_i} + b) \geq 1$

2. Non-linear SVM: Non-linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



8.4 Advantages of Support Vector Machines:

- Effective in High-Dimensional Spaces: SVMs perform well even in high-dimensional feature spaces.
- Robust to Overfitting: SVMs are less prone to overfitting, especially when the margin is maximized. Accurate for Non-Linear Data: The kernel trick allows SVMs to work effectively on non-linear data by transforming it into higher dimensions.
- Wide Applicability: SVMs can be applied to various tasks, including classification, regression, and outlier detection.
- Strong Theoretical Foundation: SVMs are based on solid mathematical principles.

8.5 Disadvantages of Support Vector Machines:

- Computationally Intensive: Training an SVM can be computationally expensive, especially for large datasets.
- Sensitivity to Kernel Choice: The choice of the kernel function and kernel parameters can significantly impact the SVM's performance.
- Challenging for Large Datasets: SVMs may not be suitable for very large datasets because of their computational complexity.
- Interpretability: The decision boundary learned by SVMs can be challenging to interpret, especially in high-dimensional spaces.

9 Bias-Variance Trade-Off

- The goal of supervised machine learning is to learn or derive a target function that can best determine the target variable from the set of input variables.
- A key consideration in learning the target function from the training data is the extent of generalization. This is because the input data is just a limited, specific view and the new, unknown data in the test data set may be differing quite a bit from the training data.
- The fitness of a target function approximated by a learning algorithm determines how correctly it is able to classify a set of data it has never seen.

9.1 Underfitting

- If the target function is kept too simple, it may not be able to capture the essential nuances and represent the underlying data well.
- A typical case of underfitting may occur when trying to represent a non-linear data with a linear model as demonstrated by both cases of underfitting shown in figure 1.1
- Many times underfitting happens due to the unavailability of sufficient training data.
- Underfitting results in both poor performance with training data as well as poor generalization to test data. Underfitting can be avoided by
  1. using more training data
  2. reducing features by effective feature selection

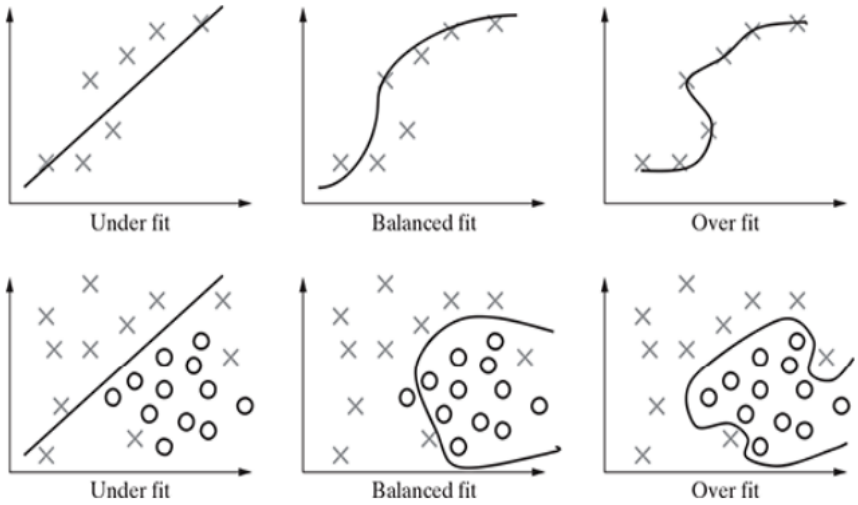


Figure 3: Figure:1.1 Underfitting and Overfitting of models

9.2 Overfitting

- Overfitting refers to a situation where the model has been designed in such a way that it emulates the training data too closely. In such a case, any specific deviation in the training data, like noise or outliers, gets embedded in the model. It adversely impacts the performance of the model on the test data.

- Overfitting, in many cases, occur as a result of trying to fit an excessively complex model to closely match the training data. This is represented with a sample data set in figure 1.1. The target function, in these cases, tries to make sure all training data points are correctly partitioned by the decision boundary. However, more often than not, this exact nature is not replicated in the unknown test data set. Hence, the target function results in wrong classification in the test data set.

- Overfitting results in good performance with training data set, but poor generalization and hence poor performance with test data set. Overfitting can be avoided by
  1. using re-sampling techniques like k-fold cross validation
  2. hold back of a validation data set
  3. remove the nodes which have little or no predictive power for the given machine learning problem.

- Both underfitting and overfitting result in poor classification quality which is reflected by low classification accuracy

9.3 Bias – variance trade-off

Bias-variance trade-off is a fundamental concept in machine learning that refers to the balance between two sources of error that affect the predictive performance of a model: bias and variance.
Bias: Bias is the error due to overly simplistic assumptions in the learning algorithm. High bias can lead to underfitting, where the model is too simple to capture the underlying patterns in the data.
Variance: Variance is the error due to too much complexity in the learning algorithm. High variance can lead to overfitting, where the model is overly sensitive to noise in the training data and fails to generalize well to new, unseen data.
In supervised learning, the class value assigned by the learning model built based on the training data may differ from the actual class value. This error in learning can be of two types – errors due to 'bias' and error due to 'variance'. Let's try to understand each of them in details.

Error due to 'Bias':

- Errors due to bias arise from simplifying assumptions made by the model to make the target function less complex or easier to learn. In short, it is due to underfitting of the model.

- Parametric models generally have high bias making them easier to understand/interpret and faster to learn.

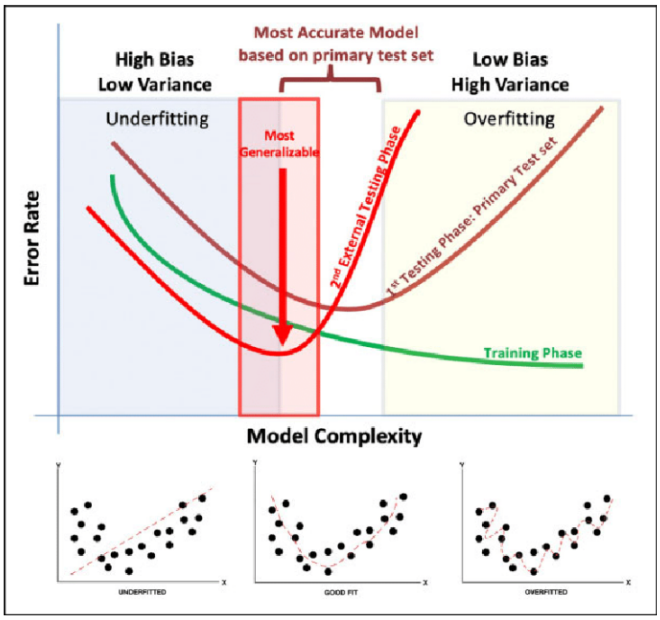
- These algorithms have a poor performance on data sets, which are complex in nature and do not align with the simplifying assumptions made by the algorithm.

- Underfitting results in high bias.

Errors due to 'Variance':

- Errors due to variance occur from difference in training data sets used to train the model.
- Different training data sets (randomly sampled from the input data set) are used to train the model. Ideally the difference in the data sets should not be significant and the model trained using different training data sets should not be too different.
- However, in case of overfitting, since the model closely matches the training data, even a small difference in training data gets magnified in the model.

So, the problems in training a model can either happen because either
(a) the model is too simple and hence fails to interpret the data grossly or
(b) the model is extremely complex and magnifies even small differences in the training data.



Key points about the bias-variance trade-off:

- **Complex Models vs. Simple Models:** Complex models (e.g., deep neural networks) tend to have low bias but high variance, whereas simple models (e.g., linear regression) tend to have high bias but low variance.
- **Balancing Act:** Machine learning practitioners aim to strike a balance between bias and variance to achieve a model with good generalization, one that performs well on both the training data and new, unseen data.
- **Underfitting and Overfitting:** The trade-off helps address the problems of underfitting (high bias) and overfitting (high variance). Underfit models don't capture enough of the data's complexity, while overfit models fit noise in the data.
- **Model Complexity:** Adjusting model complexity, such as the number of features, the choice of hyperparameters, and regularization techniques, is a way to manage the bias-variance trade-off.

- **Cross-Validation:** Cross-validation techniques, like k-fold cross-validation, help estimate a model's performance on unseen data and guide the selection of the optimal model complexity.
- **Generalization:** Achieving good generalization, where a model performs well on new, unseen data, is the ultimate goal of managing the bias-variance trade-off.

Important Note

Increasing the bias will decrease the variance, and Increasing the variance will decrease the bias On one hand, parametric algorithms are generally seen to demonstrate high bias but low variance. On the other hand, non-parametric algorithms demonstrate low bias and high variance.

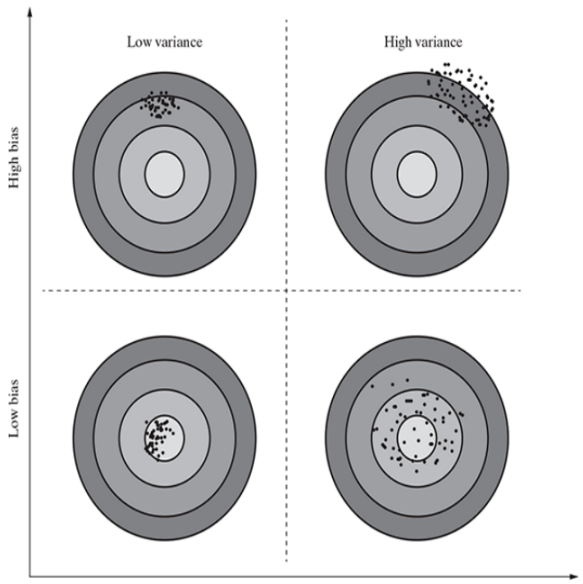


Figure 4: Figure 1.3.1

As can be observed in Figure 1.3.1, the best solution is to have a model with low bias as well as low variance. However, that may not be possible in reality. Hence, the goal of supervised machine learning is to achieve a balance between bias and variance. The learning algorithm chosen and the user parameters which can be configured helps in striking a tradeoff between bias and variance. For example, in a popular supervised algorithm k-Nearest Neighbors or kNN, the user configurable parameter 'k' can be used to do a trade-off between bias and variance. In one hand, when the value of 'k' is decreased, the model becomes simpler to fit and bias increases. On the other hand, when the value of 'k' is increased, the variance increases.

10 Cross-validation methods

- When the dataset is small, the method is prone to high variance. Due to the random partition, the results can be entirely different for different test sets. To deal with this issue, we use cross-validation to evaluate the performance of a machine-learning model.
- In cross-validation, we don't divide the dataset into training and test sets only once. Instead, we repeatedly partition the dataset into smaller groups and then average the performance in each group. That way, we reduce the impact of partition randomness on the results.
- Many cross-validation techniques define different ways to divide the dataset at hand. We'll focus on the two most frequently used: the k-fold and the leave-one-out methods.

10.1 K-Fold Cross-Validation

K-Fold Cross-Validation is a widely used technique in machine learning for assessing the performance and generalization ability of a model. It involves dividing the dataset into 'k' subsets of approximately equal size, where one of these subsets is used as the test set, and the remaining 'k-1' subsets are used as the training set. This process is repeated 'k' times, each time using a different subset as the test set.

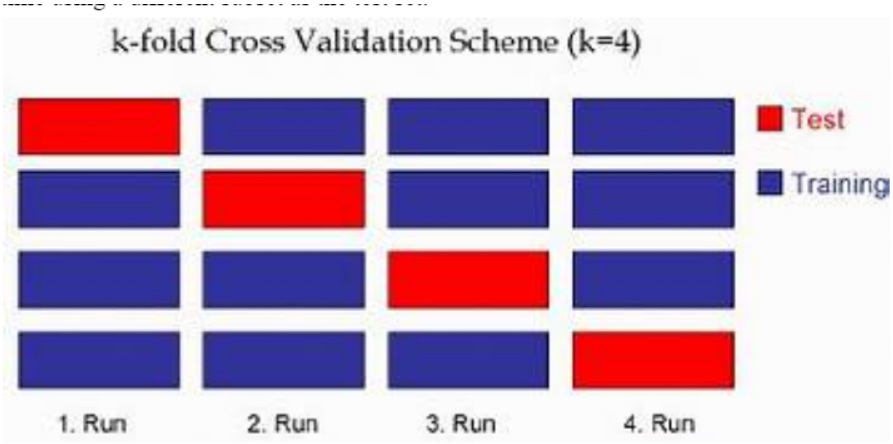


Figure 5: K-Fold Cross-Validation

Here are the key points about K-Fold Cross-Validation:

- **Data Splitting:** The dataset is divided into 'k' subsets or folds, where each fold is used as the test set exactly once, and the rest are used for training.
- **Multiple Evaluations:** K-Fold Cross-Validation enables multiple evaluations of the model's performance, providing a more reliable estimate of its generalization ability compared to a single train-test split.
- **Bias-Variance Trade-Off:** It helps in managing the bias-variance trade-off. The model's performance is assessed under different training and test subsets, helping you detect issues like overfitting or underfitting.

- **Hyperparameter Tuning:** K-Fold Cross-Validation is often used for hyperparameter tuning. By trying different hyperparameters on different folds, you can choose the set of hyperparameters that yield the best average performance.
- **K-Fold Variations:** Variations include stratified K-Fold, which ensures that each fold has a similar class distribution, and repeated K-Fold, where the process is repeated multiple times with different random splits.
- **Performance:** The final model performance is typically determined by averaging the results of all 'k' iterations, such as mean accuracy or root mean squared error.
- **Trade-Off:** There's a trade-off between computational cost and model assessment quality. Larger 'k' values lead to a more accurate assessment but require more computation.
- **Usage:** K-Fold Cross-Validation is widely used in various machine learning tasks, including model selection, hyperparameter tuning, and performance estimation.
- **Validation Set:** In practice, a separate validation set might be used to validate the final model after hyperparameter tuning, while K-Fold Cross-Validation helps in assessing the overall performance of the model.

10.2 Leave-One-Out Cross-Validation

Cross-validation methods, including leave-one-out (LOO) cross-validation, are techniques used in machine learning to assess the performance and generalization ability of a model.
In the leave-one-out (LOO) cross-validation, we train our machine-learning model n times where n is to our dataset's size. Each time, only one sample is used as a test set while the rest are used to train our model.

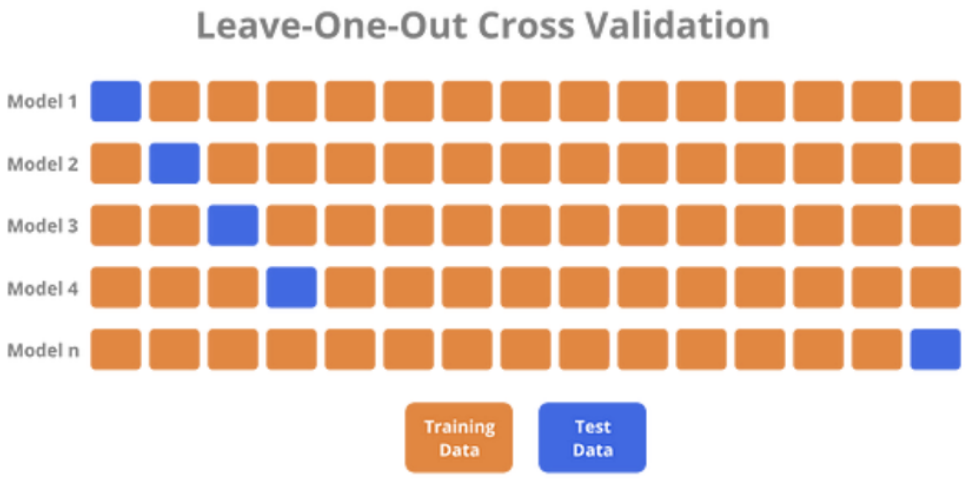


Figure 6: Leave-One-Out Cross-Validation

Here's an explanation of LOO cross-validation and its role:

- **Principle:** LOO cross-validation is a special case of k-fold cross-validation, where k is equal to the number of data points (n) in the dataset. It involves splitting the dataset into n subsets, each containing a single data point. For each iteration, one data point is held out as the test set, and the remaining n-1 data points are used as the training set. This process is repeated n times (once for each data point), and the model's performance is evaluated by averaging the results across all iterations.

- **Comprehensive Evaluation:** LOO cross-validation provides an exhaustive assessment of a model's performance, as each data point is used as a test set exactly once. This makes it suitable for small to moderately sized datasets.

- **Bias and Variance:** It tends to produce a more reliable estimate of a model's performance as it reduces bias compared to other cross-validation methods like k-fold cross-validation. However, LOO can have high variance due to its many iterations, making it computationally expensive.

- **Model Evaluation:** LOO cross-validation allows you to assess how well the model generalizes to unseen data and identify potential issues like overfitting or data leakage.

- **Advantages:** LOO is particularly useful when dealing with imbalanced datasets or when each data point is scarce and valuable, as none are left out during training.

- **Computational Cost:** LOO cross-validation can be computationally expensive, especially for large datasets, as it requires training the model n times. Variance Estimation: It's valuable for estimating the variance of performance metrics, helping you understand the stability and robustness of your model's predictions.

Comparison

An important factor when choosing between the k-fold and the LOO cross-validation methods is the size of the dataset.

When the size is small, LOO is more appropriate since it will use more training samples in each iteration. That will enable our model to learn better representations.

Conversely, we use k-fold cross-validation to train a model on a large dataset since LOO trains n models, one per sample in the data. When our dataset contains a lot of samples, training so many models will take too long. So, the k-fold cross-validation is more appropriate.

Also, in a large dataset, it is sufficient to use less than n folds since the test folds are large enough for the estimates to be sufficiently precise.

11 Feedforward Neural Network (FNN)

Neural networks are a class of machine learning models inspired by the structure and function of the human brain. Feedforward Neural Networks (FNN) represent the simplest form of neural networks, where information travels in one direction, from the input layer to the output layer.

11.1 Architecture of a Feedforward Neural Network

The architecture of a feedforward neural network consists of three types of layers: the input layer, hidden layers, and the output layer. Each layer is made up of units known as neurons, and the layers are interconnected by weights.

- **Input Layer:** This layer consists of neurons that receive inputs and pass them on to the next layer. The number of neurons in the input layer is determined by the dimensions of the input data.
- **Hidden Layers:** These layers are not exposed to the input or output and can be considered as the computational engine of the neural network. Each hidden layer's neurons take the weighted sum of the outputs from the previous layer, apply an activation function, and pass the result to the next layer. The network can have zero or more hidden layers.
- **Output Layer:** The final layer that produces the output for the given inputs. The number of neurons in the output layer depends on the number of possible outputs the network is designed to produce.
- Each neuron in one layer is connected to every neuron in the next layer, making this a fully connected network. The strength of the connection between neurons is represented by weights, and learning in a neural network involves updating these weights based on the error of the output.

The input and hidden layers use sigmoid and linear activation functions whereas the output layer uses a Heaviside step activation function at nodes because it is a two-step activation function that helps in predicting results as per requirements. All units also known as neurons have weights and calculation at the hidden layer is the summation of the dot product of all weights and their signals and finally the sigmoid function of the calculated sum. Multiple hidden and output layer increases the accuracy of the output.

11.2 Neurons,Activation Functions, Weights and Biases

- **Neurons**

  Nodes in the network that receive inputs, perform a weighted sum, and pass the result through an activation function.

- **Activation Functions**

  Non-linear functions applied to the weighted sum to introduce non-linearity and enable the network to learn complex patterns.

- **Weights**

  Parameters that the network learns during training, determining the strength of connections between neurons.

- **Biases**

  Additional parameters that are added to the weighted sum before applying the activation function, allowing the network to better fit the data.

11.3 Feedforward Process

1. The input data is fed into the input layer.
2. Each neuron in the hidden layers processes the input using weights, biases, and activation functions.
3. The output from each hidden layer is passed to the next layer.
4. This process continues until the output layer produces the final prediction.

11.4 How Feedforward Neural Networks Work

The working of a feedforward neural network involves two phases: **the feedforward phase** and the **backpropagation phase.**

- **Feedforward Phase:** In this phase, the input data is fed into the network, and it propagates forward through the network. At each hidden layer, the weighted sum of the inputs is calculated and passed through an activation function, which introduces non-linearity into the model. This process continues until the output layer is reached, and a prediction is made.
- **Backpropagation Phase:** Once a prediction is made, the error (difference between the predicted output and the actual output) is calculated. This error is then propagated back through the network, and the weights are adjusted to minimize this error. The process of adjusting weights is typically done using a gradient descent optimization algorithm.

11.5 Optimization Techniques

- **Gradient Descent**

  An iterative optimization algorithm that adjusts weights and biases to minimize the loss function.

- **Learning Rate**

  A hyperparameter that determines the step size in the weight and bias updates.

- **Mini-Batch Gradient Descent**

  An optimization technique that processes the training data in small batches to speed up convergence.

11.6 Common Activation Functions

- **Sigmoid**

  Outputs values between 0 and 1, commonly used in the output layer for binary classification.

- **Hyperbolic Tangent (tanh)**

  Similar to the sigmoid but outputs values between -1 and 1, often used in hidden layers.

- **Rectified Linear Unit (ReLU)**

  Outputs the input for positive values and zero for negative values, widely used in hidden layers.

Overfitting
When a model performs well on the training data but poorly on new, unseen data.

Regularization Techniques
Methods like dropout and L2 regularization are employed to prevent overfitting by penalizing overly complex models.

Applications of Feedforward Neural Networks
Image and speech recognition, natural language processing, financial forecasting, and many other tasks where complex patterns need to be learned from data.

12 Multi-Layer Perceptron

- A Multi-Layer Perceptron (MLP) is a class of feedforward artificial neural networks, often used in machine learning and deep learning for various tasks, such as classification, regression, and pattern recognition. An MLP consists of multiple layers of interconnected nodes, where information flows in one direction, from the input layer to the output layer, without feedback loops.
- A multi-layered perceptron consists of interconnected neurons transferring information to each other, much like the human brain. Each neuron is assigned a value. The network can be divided into three main layers.
- The MLP is a feedforward neural network, which means that the data is transmitted from the input layer to the output layer in the forward direction.
- The connections between the layers are assigned weights. The weight of a connection specifies its importance. This concept is the backbone of an MLP's learning process.
- While the inputs take their values from the surroundings, the values of all the other neurons are calculated through a mathematical function involving the weights and values of the layer before it.

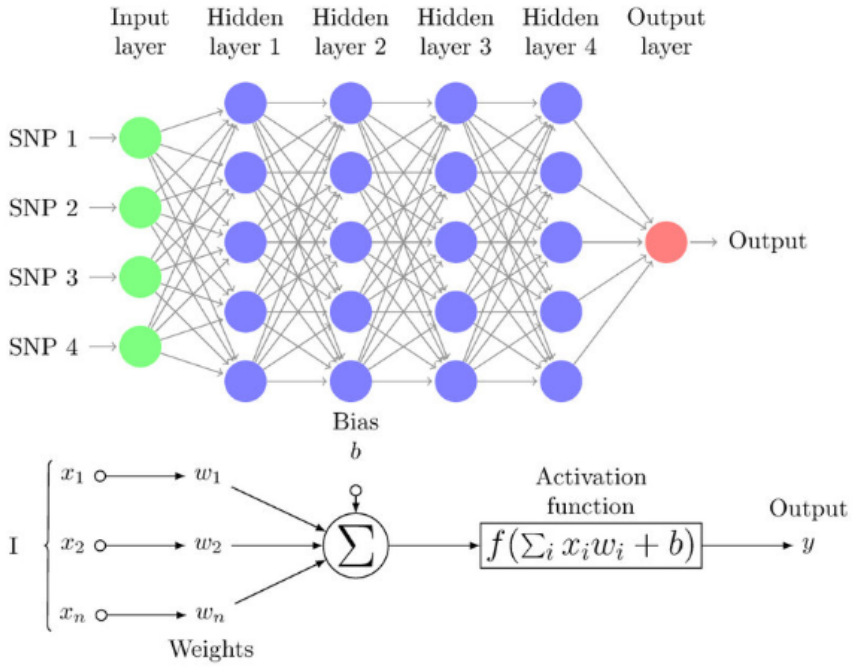


Figure 7: Multi-Layer Perceptron

12.1 Architecture:

- **Input Layer:** The input layer is responsible for receiving data from the outside world. Each neuron in the input layer corresponds to one feature, and the values from the dataset are directly fed into these neurons.

- **Hidden Layers:** Between the input and output layers, there can be one or more hidden layers. These layers contain neurons, also known as units or nodes, which are responsible for learning complex patterns and relationships in the data. Hidden layers add the capacity to model non-linear functions. An MLP can have a varying number of hidden layers and units, depending on the problem's complexity.

- **Output Layer:** The output layer is responsible for producing the final results or predictions. The number of output neurons depends on the nature of the task. For instance, in binary classification, there might be a single output neuron that outputs the probability of belonging to one class, while in multi-class classification, there could be multiple output neurons, each corresponding to a class.

12.2 Backpropagation

- Backpropagation is a technique used to optimize the weights of an MLP using the outputs as inputs.

- In a conventional MLP, random weights are assigned to all the connections. These random weights propagate values through the network to produce the actual output. Naturally, this output would differ from the expected output. The difference between the two values is called the error.

- Backpropagation refers to the process of sending this error back through the network, readjusting the weights automatically so that eventually, the error between the actual and expected output is minimized.

- In this way, the output of the current iteration becomes the input and affects the next output. This is repeated until the correct output is produced. The weights at the end of the process would be the ones on which the neural network works correctly.

12.3 Hyperparameter Tuning:

- **Architecture:** Determining the number of hidden layers, the number of neurons in each layer, and the choice of activation functions are important architectural decisions.

- **Learning Rate:**Choosing an appropriate learning rate is crucial for effective training, as it controls the size of weight updates during backpropagation.

- **Regularization:** Regularization techniques like dropout, L1, and L2 regularization are used to prevent overfitting.

13 Intro to UnSupervised Learning

- Unsupervised learning in artificial intelligence is a type of machine learning that learns from data without human supervision.
- Unlike supervised learning, unsupervised machine learning models are given unlabeled data and allowed to discover patterns and insights without any explicit guidance or instruction.
- Unsupervised learning is a machine learning paradigm where training examples lack labels, and clustering prototypes are typically initialized randomly. The primary goal is to optimize these cluster prototypes based on similarities among the training examples.
- Unsupervised learning is a machine learning paradigm that deals with unlabeled data and aims to group similar data items into clusters. It differs from supervised learning, where labeled data is used for classification or regression tasks. Unsupervised learning has applications in text clustering and other domains and can be adapted for supervised learning when necessary.
- Unsupervised learning is the opposite of supervised learning. In supervised learning, training examples are labeled with output values, and the algorithms aim to minimize errors or misclassifications. In unsupervised learning, the focus is on maximizing similarities between cluster prototypes and data items.
- Unsupervised learning doesn't refer to a specific algorithm but rather a general framework. The process involves deciding on the number of clusters, initializing cluster prototypes, and iteratively assigning data items to clusters based on similarity. These clusters are then updated until convergence is achieved.
- Unsupervised learning algorithms have various real-world applications. For instance, they can be used in text clustering, as seen in the application of algorithms like AHC and Kohonen Networks to manage and cluster textual data. They can also be used for tasks like detecting redundancy in national research projects.
- While unsupervised learning focuses on clustering, it is possible to modify unsupervised learning algorithms to function in supervised learning scenarios.

13.1 Working

As the name suggests, unsupervised learning uses self-learning algorithms—they learn without any labels or prior training. Instead, the model is given raw, unlabeled data and has to infer its own rules and structure the information based on similarities, differences, and patterns without explicit instructions on how to work with each piece of data.

Unsupervised learning algorithms are better suited for more complex processing tasks, such as organizing large datasets into clusters. They are useful for identifying previously undetected patterns in data and can help identify features useful for categorizing data.

Imagine that you have a large dataset about weather. An unsupervised learning algorithm will go through the data and identify patterns in the data points. For instance, it might group data

by temperature or similar weather patterns.
While the algorithm itself does not understand these patterns based on any previous information you provided, you can then go through the data groupings and attempt to classify them based on your understanding of the dataset. For instance, you might recognize that the different temperature groups represent all four seasons or that the weather patterns are separated into different types of weather, such as rain, sleet, or snow.

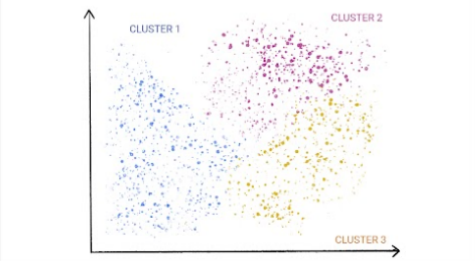


Figure 1. An ML model clustering similar data points.

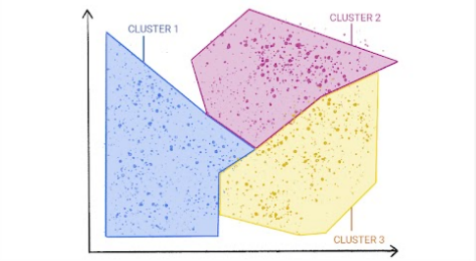


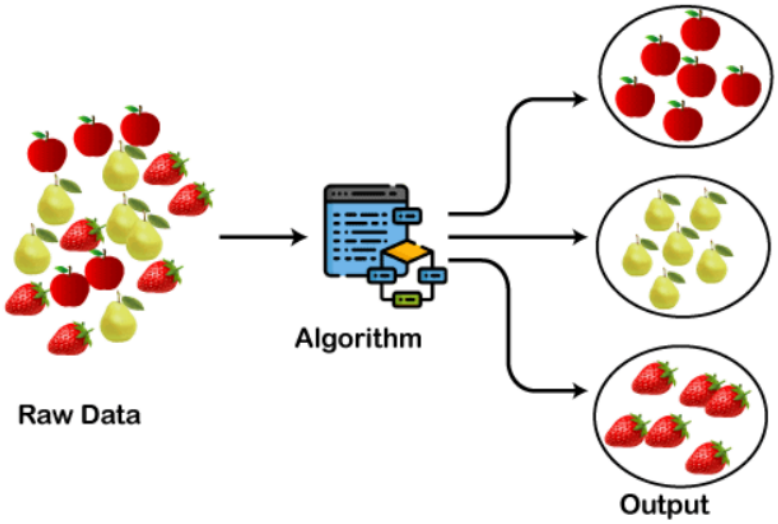
Figure 2. Groups of clusters with natural demarcations.

13.2 Unsupervised machine learning methods

In general, there are three types of unsupervised learning tasks: clustering, association rules, and dimensionality reduction.

1. **Clustering**

   Clustering is a technique for exploring raw, unlabeled data and breaking it down into groups (or clusters) based on similarities or differences. It is used in a variety of applications, including customer segmentation, fraud detection, and image analysis. Clustering algorithms split data into natural groups by finding similar structures or patterns in uncategorized data.



   Clustering is one of the most popular unsupervised machine learning approaches. There are several types of unsupervised learning algorithms that are used for clustering, which include exclusive, overlapping, hierarchical, and probabilistic.

- **Exclusive clustering:** Data is grouped in a way where a single data point can only exist in one cluster. This is also referred to as "hard" clustering. A common example of exclusive clustering is the K-means clustering algorithm, which partitions data points into a user-defined number K of clusters.
- **Overlapping clustering:** Data is grouped in a way where a single data point can exist in two or more clusters with different degrees of membership. This is also referred to as "soft" clustering.
- **Hierarchical clustering:** Data is divided into distinct clusters based on similarities, which are then repeatedly merged and organized based on their hierarchical relationships. There are two main types of hierarchical clustering: agglomerative and divisive clustering. This method is also referred to as HAC—hierarchical cluster analysis.
- **Probabilistic clustering:** Data is grouped into clusters based on the probability of each data point belonging to each cluster. This approach differs from the other methods, which group data points based on their similarities to others in a cluster.

2. **Association**

   Association rule mining is a rule-based approach to reveal interesting relationships between data points in large datasets. Unsupervised learning algorithms search for frequent if-then associations—also called rules—to discover correlations and co-occurrences within the data and the different connections between data objects.

   It is most commonly used to analyze retail baskets or transactional datasets to represent how often certain items are purchased together. These algorithms uncover customer purchasing patterns and previously hidden relationships between products that help inform recommendation engines or other cross-selling opportunities. You might be most familiar with these rules from the "Frequently bought together" and "People who bought this item also bought" sections on your favorite online retail shop.

   Association rules are also often used to organize medical datasets for clinical diagnoses. Using unsupervised machine learning and association rules can help doctors identify the probability of a specific diagnosis by comparing relationships between symptoms from past patient cases.

   Typically, Apriori algorithms are the most widely used for association rule learning to identify related collections of items or sets of items. However, other types are used, such as Eclat and FP-growth algorithms.

3. **Dimensionality reduction**

   Dimensionality reduction is an unsupervised learning technique that reduces the number of features, or dimensions, in a dataset. More data is generally better for machine learning, but it can also make it more challenging to visualize the data.

   Dimensionality reduction extracts important features from the dataset, reducing the number of irrelevant or random features present. This method uses principle component analysis (PCA) and singular value decomposition (SVD) algorithms to reduce the number of data inputs without compromising the integrity of the properties in the original data.

13.3 Supervised learning vs. unsupervised learning

The main difference between supervised learning and unsupervised learning is the type of input data that you use. Unlike unsupervised machine learning algorithms, supervised learning relies on labeled training data to determine whether pattern recognition within a dataset is accurate.

The goals of supervised learning models are also predetermined, meaning that the type of output of a model is already known before the algorithms are applied. In other words, the input is mapped to the output based on the training data.

Supervised Learning	**Unsupervised Learning**
Supervised learning algorithms are trained using labeled data.	Unsupervised learning algorithms are trained using unlabeled data.
Supervised learning model takes direct feedback to check if it is predicting correct output or not.	Unsupervised learning model does not take any feedback.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
The goal of supervised learning is to train the model so that it can predict the output when it is given new data.	The goal of unsupervised learning is to find the hidden patterns and useful insights from the unknown dataset.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.
Supervised learning can be categorized in Classification and Regression problems.	Unsupervised Learning can be classified in Clustering and Associations problems.
Supervised learning can be used for those cases where we know the input as well as corresponding outputs.	Unsupervised learning can be used for those cases where we have only input data and no corresponding output data.
Supervised learning model produces an accurate result.	Unsupervised learning model may give less accurate result as compared to supervised learning.
Supervised learning is not close to true Artificial intelligence as in this, we first train the model for each data, and then only it can predict the correct output.	Unsupervised learning is more close to the true Artificial Intelligence as it learns similarly as a child learns daily routine things by his experiences.
It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	It includes various algorithms such as Clustering, KNN, and Apriori algorithm.

13.4 Types of Clustering

Clustering is a type of unsupervised learning wherein data points are grouped into different sets based on their degree of similarity.

The various types of clustering are:

- **Hierarchical clustering**
- **Partitioning clustering**

Hierarchical clustering is further subdivided into:

- Agglomerative clustering
- Divisive clustering

Partitioning clustering is further subdivided into:

- K-Means clustering
- Fuzzy C-Means clustering

There are many different fields where cluster analysis is used effectively, such as

- **Text data mining:** this includes tasks such as text categorization, text clustering, document summarization, concept extraction, sentiment analysis, and entity relation modelling
- **Customer segmentation:** creating clusters of customers on the basis of parameters such as demographics, financial conditions, buying habits, etc., which can be used by retailers and advertisers to promote their products in the correct segment
- **Anomaly checking:** checking of anomalous behaviours such as fraudulent bank transaction, unauthorized computer intrusion, suspicious movements on a radar scanner, etc.
- **Data mining:** simplify the data mining task by grouping a large number of features from an extremely large data set to make the analysis manageable

14 K-Means Clustering

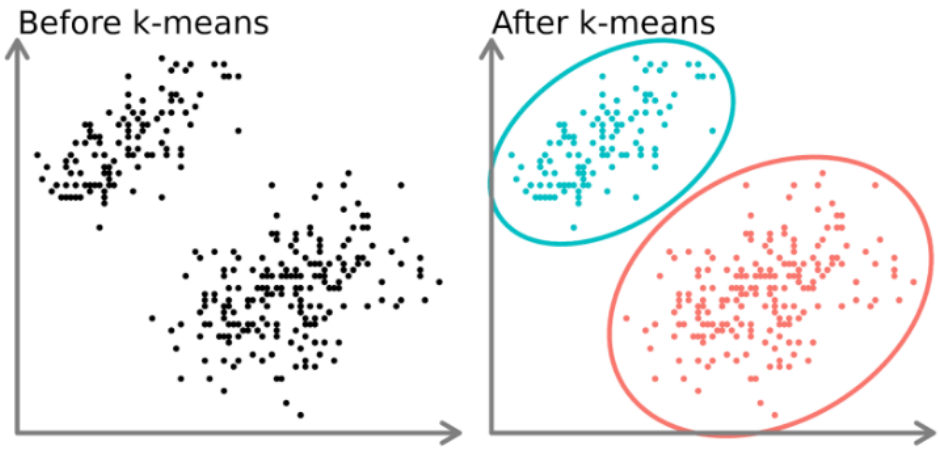
K-Medoids and K-Means are two types of clustering mechanisms in Partition Clustering. First, Clustering is the process of breaking down an abstract group of data points/ objects into classes of similar objects such that all the objects in one cluster have similar traits. , a group of **n** objects is broken down into **k** number of clusters based on their similarities.

- The K-Means algorithm is a popular clustering algorithm used in unsupervised machine learning to partition a dataset into K distinct, non-overlapping clusters.
- It aims to find cluster centers (centroids) and assign data points to the nearest centroid based on their similarity.
- K-means clustering is one of the simplest and popular unsupervised machine learning algorithms.
- Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.
- A cluster refers to a collection of data points aggregated together because of certain similarities.
- K-means is a very simple to implement clustering algorithm that works by selecting k centroids initially, where k serves as the input to the algorithm and can be defined as the number of clusters required. The centroids serve as the center of each new cluster.
- We first assign each data point of the given data to the nearest centroid. Next, we calculate the mean of each cluster and the means then serve as the new centroids. This step is repeated until the positions of the centroids do not change anymore.
- The goal of k-means is to minimize the sum of the squared distances between each data point and its centroid.
- The algorithm aims to minimize the sum of squared distances between data points and their respective cluster centroids.

$$J = \sum_{i=1}^{K} \sum_{j=1}^{n_i} \|x_{ij} - c_i\|^2$$

where:

K is the number of clusters,
n_i is the number of data points in cluster i,
x_{ij} is the j-th data point in cluster i,
c_i is the centroid of cluster i.



K-means algorithm – (cont …)

Algorithm *k*-means(*k*, *D*)

1 Choose *k* data points as the initial centroids (cluster centers)
2 **repeat**
3 **for** each data point $\mathbf{x} \in D$ do
4 compute the distance from **x** to each centroid;
5 assign **x** to the closest centroid // a centroid represents a cluster
6 **endfor**
7 re-compute the centroids using the current cluster memberships
8 **until** the stopping criterion is met

K-means algorithm – (cont …)

Algorithm *k*-means(*k*, *D*)

1 Choose *k* data points as the initial centroids (cluster centers)
2 **repeat**
3 **for** each data point $\mathbf{x} \in D$ do
4 compute the distance from **x** to each centroid;
5 assign **x** to the closest centroid // a centroid represents a cluster
6 **endfor**
7 re-compute the centroids using the current cluster memberships
8 **until** the stopping criterion is met

Stopping/convergence criterion

1. no (or minimum) re-assignments of data points to different clusters,
2. no (or minimum) change of centroids, or
3. minimum decrease in the **sum of squared error** (SSE),

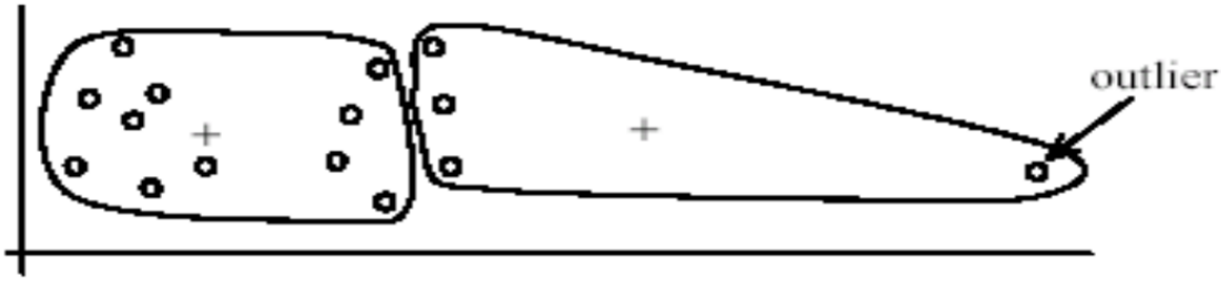
$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2 \quad (1)$$

– C_i is the jth cluster, $\mathbf{m}_j$ is the centroid of cluster C_j (the mean vector of all the data points in C_j), and $dist(\mathbf{x}, \mathbf{m}_j)$ is the distance between data point **x** and centroid $\mathbf{m}_j$.

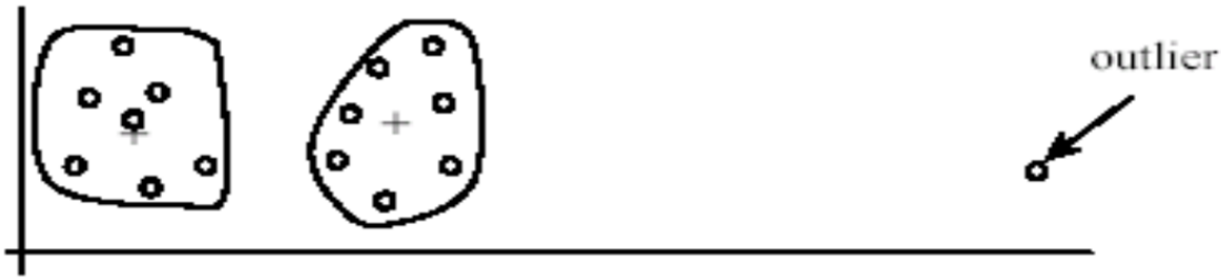
- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: $O(tkn)$,
    where n is the number of data points,
    k is the number of clusters, and
    t is the number of iterations.
  - Since both k and t are small. k-means is considered a linear algorithm.
- K-means is the most popular clustering algorithm.
- Note that: it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

- The algorithm is only applicable if the mean is defined.
  - For categorical data, k-mode - the centroid is represented by most frequent values.
- The user needs to specify k.
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



(B): Ideal clusters

Here's an overview of the K-Means algorithm:

- **Initialization:** Choose the number of clusters (K) you want to create. Initialize K cluster centroids randomly. These centroids can be selected from the data points or with random values.
- **Assignment Step:** For each data point in the dataset, calculate the distance between the data point and all K centroids. Assign the data point to the cluster associated with the nearest centroid. This step groups data points into clusters based on similarity.
- **Update Step:** Recalculate the centroids for each cluster by taking the mean of all data points assigned to that cluster. The new centroids represent the center of their respective clusters.
- **Convergence Check:** Check if the algorithm has converged. Convergence occurs when the centroids no longer change significantly between iterations. If convergence is not reached, return to the Assignment and Update steps.
- **Termination:** Once convergence is achieved or a predetermined number of iterations are completed, the algorithm terminates. The final cluster assignments and centroids are obtained.

- **Results:** The result of the K-Means algorithm is K clusters, each with its centroid. Data points are assigned to the nearest cluster, and you can use these clusters for various purposes, such as data analysis, segmentation, or pattern recognition.

Strengths	Weaknesses
• The principle used for identifying the clusters is very simple and involves very less complexity of statistical terms • The algorithm is very flexible and thus can be adjusted for most scenarios and complexities • The performance and efficiency are very high and comparable to those of any sophisticated algorithm in term of dividing the data into useful clusters	• The algorithm involves an element of random chance and thus may not find the optimal set of cluster in some cases • The starting point of guessing the number natural clusters within the data requires some experience of the user, so that the final outcome is efficient

Figure 8: Strength and Weakness of K-means

Within-Cluster Sum of Squares (WCSS) in K-means Clustering

Within-Cluster Sum of Squares (WCSS) is a metric used to evaluate the quality of the clusters formed by the K-means clustering algorithm. It measures the sum of squared distances between each data point and the centroid of the cluster to which it belongs. A lower WCSS value indicates that the data points are closer to their respective cluster centroids, which means the clusters are more compact and better defined.

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left\| x_n - \mu_k \right\|^2$$

Purpose of WCSS

- Minimization of WCSS is the goal of the K-means algorithm. The algorithm iteratively tries to adjust the positions of the centroids to minimize the WCSS.
- WCSS is also used to determine the optimal number of clusters using the Elbow Method, which helps find a balance between the number of clusters and how well the data is clustered.

]

14.1 Methods to Determine the Optimal Number of Clusters (K) in K-means Clustering

In K-means clustering, the number of clusters K needs to be specified before the algorithm runs. Choosing the right value for K is crucial because it affects how well the data is clustered. Two popular methods to determine the optimal number of clusters are:

1. The Elbow Method
2. The Silhouette Method

1. The Elbow Method

The Elbow Method is one of the most commonly used techniques to determine the optimal number of clusters in K-means clustering. It is based on the idea that increasing the number of clusters K will reduce the Within-Cluster Sum of Squares (WCSS), but after a certain point, the improvement will diminish, forming an "elbow" in the curve.

Steps:

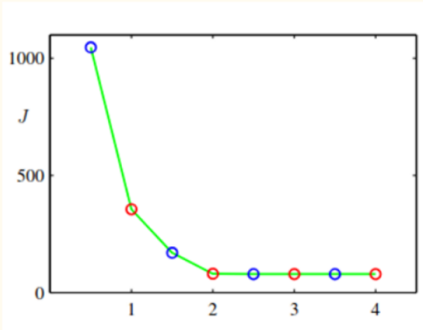
1. **Run K-means for a range of** K values (e.g., from 1 to 10).
2. **Calculate the WCSS** for each K.
   - **WCSS (Within-Cluster Sum of Squares)** is the sum of squared distances between each data point and its closest centroid. It is a measure of how compact the clusters are.
   - As K increases, WCSS will decrease, because the clusters become smaller and more tightly packed.
3. **Plot the WCSS** against K.
4. **Identify the "elbow"** point where the WCSS starts decreasing at a slower rate. This is the optimal K, as it represents the point where adding more clusters does not significantly improve the clustering.

How to Identify the Elbow:

The "elbow" is where the plot starts to bend or flatten out. This point indicates the optimal number of clusters.

If the curve has no clear elbow, choosing the right number of clusters can be subjective.

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$



Plot of the cost function J given by (9.1) after each E step (blue points) and M step (red points) of the K-means algorithm for the example shown in Figure 9.1.

2. The Silhouette Method

The Silhouette Method measures how similar a data point is to its own cluster (cohesion) compared to other clusters (separation). It computes the silhouette coefficient for each point, which quantifies how well a point fits into its assigned cluster. The average silhouette score across all points can be used to evaluate different values of K.

Silhouette Coefficient:

For each data point, the silhouette coefficient $S(i)$ is defined as:

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Where:

- $a(i)$ is the average distance of point i to all other points in its own cluster.
- $b(i)$ is the average distance of point i to the points in the nearest neighboring cluster (the next closest cluster to which it does not belong).

The **silhouette coefficient** ranges from -1 to 1:

- **1**: The data point is very well clustered (far from other clusters).
- **0**: The data point is on or very close to the decision boundary between two clusters.
- **Negative values**: The data point is likely assigned to the wrong cluster.

Steps:

1. Run K-means for a range of K values (e.g., 2 to 10).
2. Calculate the silhouette coefficient for each data point.
3. Compute the **average silhouette score** for all points for each K.
4. The **optimal** K is the one that maximizes the average silhouette score.

14.2 Comparison of Elbow and Silhouette Methods

Aspect	Elbow Method	Silhouette Method
Metric	WCSS (Within-Cluster Sum of Squares)	Silhouette Coefficient (based on cohesion and separation)
Interpretation	Look for the "elbow" in the curve where WCSS starts flattening	Look for the K with the highest average silhouette score
Optimal Number of Clusters	May not always be clear if the elbow is not prominent	Optimal K is the one with the highest silhouette score
Complexity	Simple and fast	More computationally expensive
Applicability	Works best when clusters are compact and well-separated	Works well even when clusters are of varying shapes and sizes

14.3 K-medoids algorithm

- K-medoids is an unsupervised method with unlabelled data to be clustered. It is an improvised version of the K-Means algorithm mainly designed to deal with outlier data sensitivity.
- Compared to other partitioning algorithms, the algorithm is simple, fast, and easy to implement.
- K-medoids clustering method but unlike k-means, rather than minimizing the sum of squared distances, k-medoids works on minimizing the number of paired dissimilarities.
- We find this useful since k-medoids aims to form clusters where the objects within each cluster are more similar to each other and dissimilar to objects in the other clusters. Instead of centroids, this approach makes use of medoids.
- **Medoids** are points in the dataset whose sum of distances to other points in the cluster is minimal.

K-Medoids Algorithm (Pseudocode)

Here's a simplified pseudocode for the K-medoids algorithm:

1. Select K random points from the dataset as the initial medoids.
2. **Repeat** until convergence or a maximum number of iterations:
    - For each data point:
        - Assign the data point to the nearest medoid based on a distance metric.
    - For each cluster:
        - Identify the point that minimizes the total distance to all other points in the cluster as the new medoid.
    - Update the medoids.
3. **End** when the medoids do not change between iterations.

There are three types of algorithms for K-Medoids Clustering: *(No need to go in detailed)*

1. PAM (Partitioning Around Clustering)
2. CLARA (Clustering Large Applications)
3. CLARANS (Randomized Clustering Large Applications)

PAM is the most powerful algorithm of the three algorithms but has the disadvantage of time complexity. The following K-Medoids are performed using PAM. In the further parts, we'll see what CLARA and CLARANS are.

15 Hierarchical clustering

Till now, we have discussed the various methods for partitioning the data into different clusters. But there are situations when the data needs to be partitioned into groups at different levels such as in a hierarchy. The hierarchical clustering methods are used to group the data into hierarchy or tree-like structure.
For example, in a machine learning problem of organizing employees of a university in different departments, first the employees are grouped under the different departments in the university, and then within each department, the employees can be grouped according to their roles such as professors, assistant professors, supervisors, lab assistants, etc. This creates a hierarchical structure of the employee data and eases visualization and analysis. Similarly, there may be a data set which has an underlying hierarchy structure that we want to discover and we can use the hierarchical clustering methods to achieve that.

- Hierarchical clustering (also called hierarchical cluster analysis or HCA) is a method of cluster analysis which seeks to build a hierarchy of clusters (or groups) in a given dataset.
- The hierarchical clustering produces clusters in which the clusters at each level of the hierarchy are created by merging clusters at the next lower level.
- At the lowest level, each cluster contains a single observation. At the highest level there is only one cluster containing all of the data.
- The decision regarding whether two clusters are to be merged or not is taken based on the measure of dissimilarity between the clusters. The distance between two clusters is usually taken as the measure of dissimilarity between the clusters.
- Hierarchical clustering is more interpretable than other clustering techniques because it provides a full hierarchy of clusters.
- The choice of linkage method and distance metric can significantly impact the results and the structure of the dendrogram.
- Dendrograms are useful for visualizing the hierarchy and helping to decide how many clusters are appropriate for a particular application.
- Hierarchical clustering can be computationally intensive, especially for large datasets, and may not be suitable for such cases.

15.1 Hierarchical Clustering Methods

There are two main hierarchical clustering methods: **agglomerative clustering** and **divisive clustering.**
Agglomerative clustering is a bottom-up technique which starts with individual objects as clusters and then iteratively merges them to form larger clusters. On the other hand, the **divisive method** starts with one cluster with all given objects 2 and then splits it iteratively to form smaller clusters.
In both these cases, it is important to select the split and merger points carefully, because the subsequent splits or mergers will use the result of the previous ones and there is no option to perform any object swapping between the clusters or rectify the decisions made in previous steps, which may result in poor clustering quality at the end.

A dendrogram is a commonly used tree structure representation of step-by-step creation of hierarchical clustering. It shows how the clusters are merged iteratively (in the case of agglomerative clustering) or split iteratively (in the case of divisive clustering) to arrive at the optimal clustering solution.

15.1.1 Dendrogram

- Hierarchical clustering can be represented by a rooted binary tree. The nodes of the trees represent groups or clusters. The root node represents the entire data set. The terminal nodes each represent one of the individual observations (singleton clusters). Each nonterminal node has two daughter nodes.
- The distance between merged clusters is monotone increasing with the level of the merger. The height of each node above the level of the terminal nodes in the tree is proportional to the value of the distance between its two daughters (see Figure 13.9).
- A dendrogram is a tree diagram used to illustrate the arrangement of the clusters produced by hierarchical clustering.
- The dendrogram may be drawn with the root node at the top and the branches growing vertically downwards (see Figure 13.8(a)).
- It may also be drawn with the root node at the left and the branches growing horizontally rightwards (see Figure 13.8(b)).
- In some contexts, the opposite directions may also be more appropriate.
- Dendrograms are commonly used in computational biology to illustrate the clustering of genes or samples.

Example Figure 13.7 is a dendrogram of the dataset $\{a, b, c, d, e\}$. Note that the root node represents the entire dataset and the terminal nodes represent the individual observations. However, the dendrograms are presented in a simplified format in which only the terminal nodes (that is, the nodes representing the singleton clusters) are explicitly displayed. Figure 13.8 shows the simplified format of the dendrogram in Figure 13.7. Figure 13.9 shows the distances of the clusters at the various levels. Note that the clusters are at 4 levels. The distance between the clusters $\{a\}$ and

$\{b\}$ is 15, between $\{c\}$ and $\{d\}$ is 7.5, between $\{c, d\}$ and $\{e\}$ is 15 and between $\{a, b\}$ and $\{c, d, e\}$ is 25.

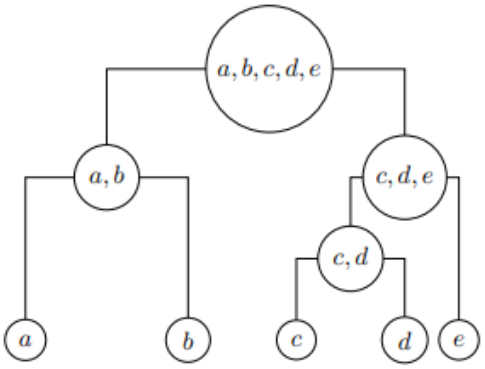


Figure 13.7: A dendrogram of the dataset $\{a, b, c, d, e\}$

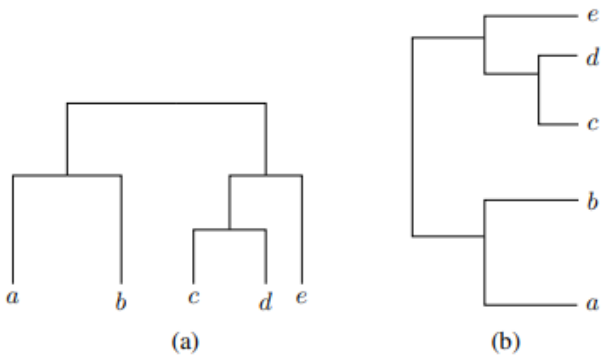


Figure 13.8: Different ways of drawing dendrogram

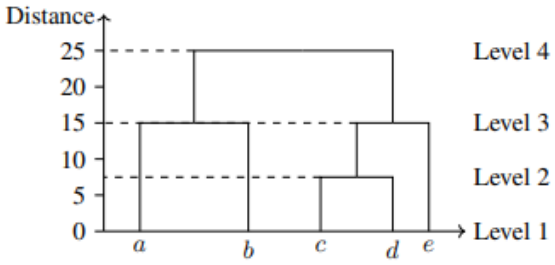
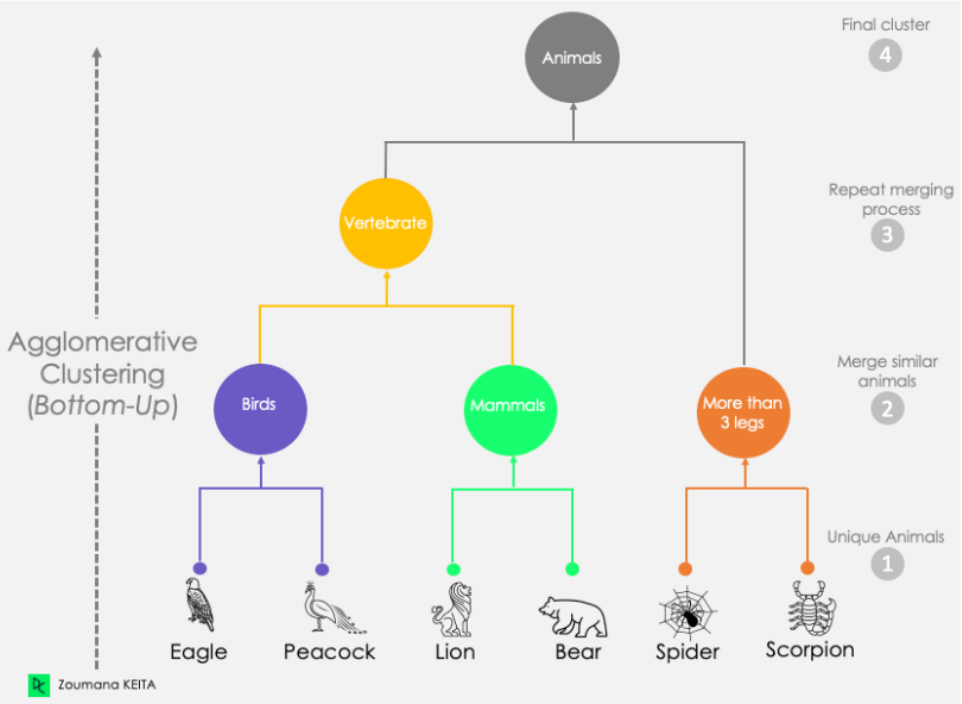


Figure 9: Figure 13.9: A dendrogram of the dataset a, b, c, d, e showing the distances (heights) of the clusters at different levels

15.1.2 Agglomerative Hierarchical Clustering (Bottom-Up)

The agglomerative hierarchical clustering method uses the bottom-up strategy. It starts with each object forming its own cluster and then iteratively merges the clusters according to their similarity to form larger clusters. It terminates either when a certain clustering condition imposed by the user is achieved or all the clusters merge into a single cluster.



If there are N observations in the dataset, there will be $N - 1$ levels in the hierarchy. The pair chosen for merging consists of the two groups with the smallest "intergroup dissimilarity". Each nonterminal node has two daughter nodes. The daughters represent the two groups that were merged to form the parent.

Here's a step-by-step explanation of the agglomerative hierarchical clustering algorithm:

- Step 1: Initialization Start with each data point as a separate cluster. If you have N data points, you initially have N clusters.
- Step 2: Merge Clusters Calculate the pairwise distances (similarity or dissimilarity) between all clusters. Common distance metrics include Euclidean distance, Manhattan distance, or other similarity measures. Merge the two closest clusters based on the distance metric. There are various linkage methods to define cluster distance:
  - **Single Linkage:** Merge clusters based on the minimum distance between any pair of data points from the two clusters.
  - **Complete Linkage:** Merge clusters based on the maximum distance between any pair of data points from the two clusters.
  - **Average Linkage:** Merge clusters based on the average distance between data points in the two clusters.
  - **Ward's Method:** Minimize the increase in variance when merging clusters. Repeat this merging process iteratively until all data points are in a single cluster or until you reach the desired number of clusters.

- Step 3: Dendrogram During the merging process, create a dendrogram, which is a tree-like structure that represents the hierarchy of clusters. The dendrogram provides a visual representation of how clusters merge and shows the relationships between clusters at different levels of granularity.

- Step 4: Cutting the Dendrogram To determine the number of clusters, you can cut the dendrogram at a specific level. The height at which you cut the dendrogram corresponds to the number of clusters you obtain. The cut produces the final clusters at the chosen level of granularity.

- Step 5: Results The resulting clusters are obtained based on the cut level. Each cluster contains a set of data points that are similar to each other according to the chosen linkage method.

For example, the hierarchical clustering shown in Figure 13.7 can be constructed by the agglomerative method as shown in Figure 13.10. Each nonterminal node has two daughter nodes. The daughters represent the two groups that were merged to form the parent.

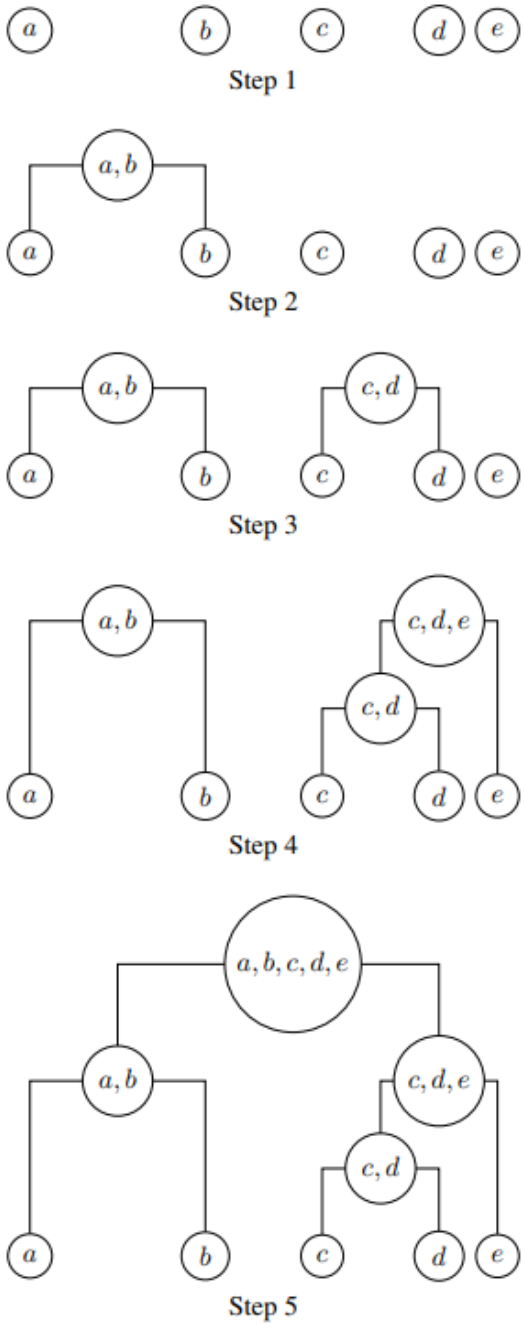


Figure 10: Figure 13.10: Hierarchical clustering using agglomerative method

15.1.3 Divisive Hierarchical Clustering (Top-Down)

- Top-down hierarchical clustering, also known as divisive hierarchical clustering, is a clustering algorithm that starts with all data points in a single cluster and recursively divides clusters into smaller sub-clusters until each data point forms its own individual cluster or a specified number of clusters is reached.
- The divisive hierarchical clustering method uses a top-down strategy. The starting point is the largest cluster with all the objects in it, and then, it is split recursively to form smaller and smaller clusters, thus forming the hierarchy. The end of iterations is achieved when the objects in the final clusters are sufficiently homogeneous to each other or the final clusters contain only one object or the user-defined clustering condition is achieved.
- The divisive method starts at the top and at each level recursively split one of the existing clusters at that level into two new clusters.
- If there are N observations in the dataset, there the divisive method also will produce $N-1$ levels in the hierarchy. The split is chosen to produce two new groups with the largest "between-group dissimilarity".

  Here's a step-by-step explanation of the divisive hierarchical clustering algorithm:

  - Step 1: Initialization

    Start with all data points as members of a single cluster. If you have N data points, you initially have one cluster with N members.
  - Step 2: Split Clusters
    - Calculate the within-cluster variance or a similar measure for the current cluster. This measure represents the compactness of the cluster.
    - Divide the cluster into two sub-clusters in a way that minimizes the within-cluster variance. There are various methods to achieve this, such as k-means or hierarchical clustering on the sub-cluster.
    - Repeat the splitting process recursively for each sub-cluster until you reach the desired number of clusters.
  - Step 3: Dendrogram (Optional)

    While divisive hierarchical clustering often doesn't produce a dendrogram like agglomerative clustering, you can still record the hierarchy of cluster splits for analysis if needed.
  - Step 4: Results

    The resulting clusters are obtained based on the recursive splits. Each cluster contains a set of data points that are similar to each other according to the splitting criteria.

For example, the hierarchical clustering shown in Figure 13.7 can be constructed by the divisive method as shown in Figure 13.11. Each nonterminal node has two daughter nodes. The two daughters represent the two groups resulting from the split of the parent.

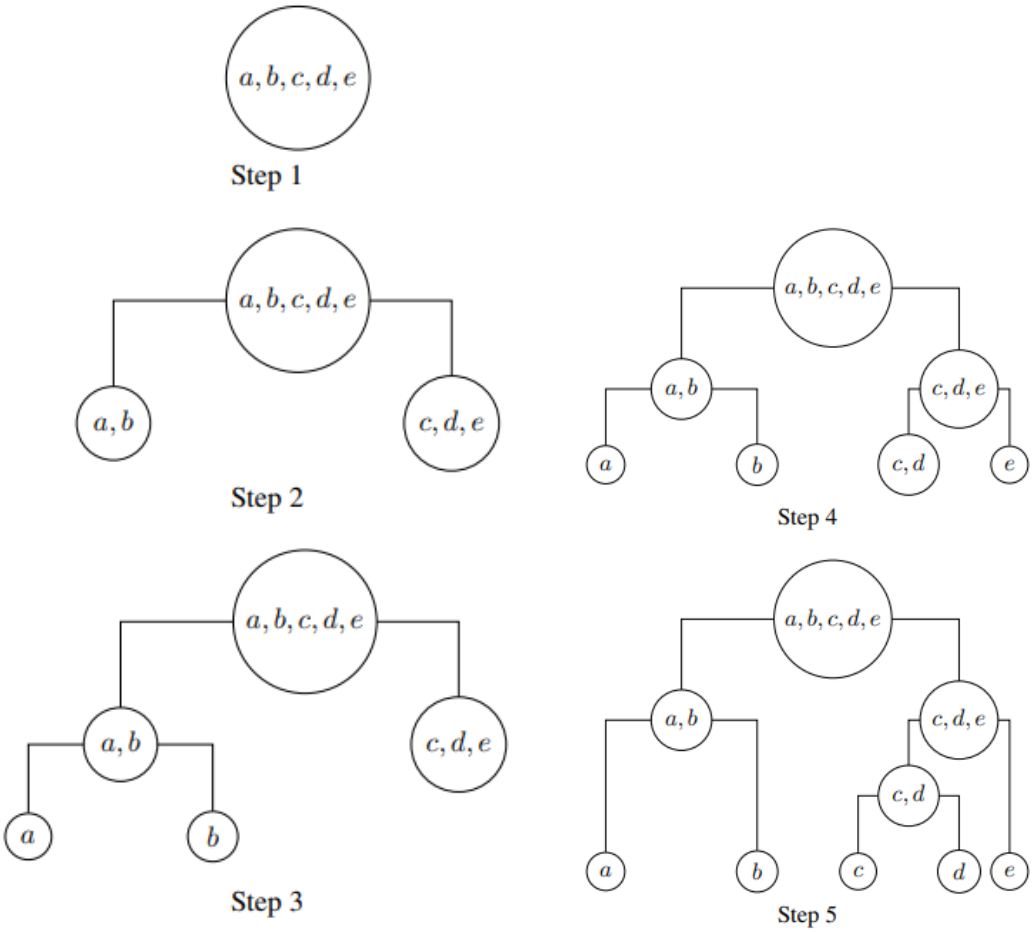


Figure 11: Figure 13.11: Hierarchical clustering using divisive method

In both these cases, it is important to select the split and merger points carefully, because the subsequent splits or mergers will use the result of the previous ones and there is no option to perform any object swapping between the clusters or rectify the decisions made in previous steps, which may result in poor clustering quality at the end.

15.2 Measures of dissimilarity

In order to decide which clusters should be combined (for agglomerative), or where a cluster should be split (for divisive), a measure of dissimilarity between sets of observations is required. In most methods of hierarchical clustering, the dissimilarity between two groups of observations is measured by using an appropriate measure of distance between the groups of observations. The distance between two groups of observations is defined in terms of the distance between two observations. There are several ways in which the distance between two observations can be defined and also there are also several ways in which the distance between two groups of observations can be defined

15.2.1 Measures of distance between data points

One of the core measures of proximities between clusters is the distance between them. There are four standard methods to measure the distance between clusters: Let C_i and C_j be the two clusters

with n_i and n_j respectively. p_i and p_j represents the points in clusters C_i and C_j respectively. We will denote the mean of cluster C_i as m_i.

$$\text{Minimum distance } D_{\min}(C_i, C_j) = \min_{p_i \in C_i,\, p_j \in C_j} \{|p_i - p_j|\}$$

$$\text{Maximum distance } D_{\max}(C_i, C_j) = \max_{p_i \in C_i,\, p_j \in C_j} \{|p_i - p_j|\}$$

$$\text{Mean distance } D_{\text{mean}}(C_i, C_j) = \{|m_i - m_j|\}$$

$$\text{Average distance } D_{\text{avg}}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p_i \in C_i,\, p_j \in C_j} |pi - pj|$$

Figure 12: Distance measure in algorithmic methods

Often the distance measure is used to decide when to terminate the clustering algorithm. For example, in an agglomerative clustering, the merging iterations may be stopped once the MIN distance between two neighboring clusters becomes less than the user-defined threshold.

So, when an algorithm uses the minimum distance D_{min} to measure the distance between the clusters, then it is referred to as nearest neighbour clustering algorithm, and if the decision to stop the algorithm is based on a user-defined limit on D_{min} , then it is called a **single linkage algorithm.**

On the other hand, when an algorithm uses the maximum distance D_{max} to measure the distance between the clusters, then it is referred to as furthest neighbour clustering algorithm, and if the decision to stop the algorithm is based on a userdefined limit on D_{max} then it is called **complete linkage algorithm.**

As minimum and maximum measures provide two extreme options to measure distance between the clusters, they are prone to the outliers and noisy data. Instead, the use of mean and average distance helps in avoiding such problem and provides more consistent results.

15.3 Single linkage

Single linkage, also known as single-link clustering, is a hierarchical agglomerative clustering method used in unsupervised machine learning. It is one of the linkage methods used in agglomerative hierarchical clustering. Single linkage defines the distance between two clusters as the minimum distance between any two data points, one from each cluster.

Single linkage works:

1. **Initialization:** Start with each data point as an individual cluster. If you have N data points, you initially have N clusters.
2. **Cluster Distance:** Calculate the pairwise distances between all clusters. The distance between two clusters is defined as the minimum distance between any two data points, one from each cluster.
3. **Merge Clusters:** Merge the two clusters with the shortest distance, as defined by single linkage. This creates a new, larger cluster.
4. **Repeat:** Continue steps 2 and 3 iteratively until all data points are part of a single cluster, or you reach a specified number of clusters.
5. **Dendrogram:** During the process, create a dendrogram, which is a tree-like structure that represents the hierarchy of clusters. It records the sequence of cluster mergers.
6. **Cutting the Dendrogram:** To determine the number of clusters, you can cut the dendrogram at a specific level. The height at which you cut the dendrogram corresponds to the number of clusters you obtain.

Single linkage has some characteristics:

- It is sensitive to outliers and noise because a single close pair of points from different clusters can cause a merger.
- It tends to create elongated clusters, as it connects clusters based on single, nearest neighbors.
- It is fast and can handle large datasets, making it computationally efficient.
- Single linkage is just one of several linkage methods used in hierarchical clustering, each with its own strengths and weaknesses. The choice of linkage method depends on the nature of the data and the desired clustering outcome.

15.4 Multiple Linkage

Multiple linkage, also known as complete linkage or maximum linkage, is a hierarchical clustering method used in agglomerative hierarchical clustering. It defines the distance between two clusters as the maximum distance between any two data points, one from each cluster. In contrast to single linkage, which uses the minimum distance, complete linkage aims to minimize the maximum distance between any two data points within the merged clusters.

Multiple linkage works:

1. **Initialization:** Start with each data point as an individual cluster. If you have N data points, you initially have N clusters.
2. **Cluster Distance:** Calculate the pairwise distances between all clusters. The distance between two clusters is defined as the maximum distance between any two data points, one from each cluster.

3. **Merge Clusters:** Merge the two clusters with the shortest (maximum) distance, as defined by complete linkage. This creates a new, larger cluster.

4. **Repeat:** Continue steps 2 and 3 iteratively until all data points are part of a single cluster or you reach a specified number of clusters.

5. **Dendrogram:** Create a dendrogram to represent the hierarchy of cluster mergers. The dendrogram records the sequence of cluster merges.

6. **Cutting the Dendrogram:** To determine the number of clusters, you can cut the dendrogram at a specific level. The height at which you cut the dendrogram corresponds to the number of clusters you obtain.

Multiple linkage has some characteristics:

- It tends to produce compact, spherical clusters since it minimizes the maximum distance within clusters.
- It is less sensitive to outliers than single linkage because it focuses on the maximum distance.
- It can handle elongated or irregularly shaped clusters effectively.

The choice of linkage method (single, complete, average, etc.) depends on the nature of the data and the desired clustering outcome. Different linkage methods can produce different cluster structures based on how distance is defined between clusters.

16 Dimensionality Reduction

Dimensionality reduction is a technique used in machine learning and data analysis to reduce the number of features (dimensions) in a dataset. It involves transforming a high-dimensional dataset into a lower-dimensional representation while retaining the most relevant information. Dimensionality reduction is useful for several reasons:

- **Curse of Dimensionality:** High-dimensional data can suffer from the curse of dimensionality, where the data becomes sparse and noisy, making it challenging to analyze and model effectively. Reducing dimensionality can help mitigate this issue.
- **Computation and Storage:** High-dimensional data requires more computational resources and storage. Reducing dimensionality can make algorithms faster and more memory-efficient.
- **Visualization:** It is difficult to visualize data in more than three dimensions. Dimensionality reduction techniques can project data into a lower-dimensional space for visualization purposes.
- **Feature Engineering:** Dimensionality reduction can be a form of automated feature selection, helping to identify the most important features and reducing the risk of overfitting.

Curse of Dimensionality

- Why are more features bad?
  - Redundant features (not all words are useful to classify a document) more noise added than signal
  - Hard to interpret and visualize
  - Hard to store and process data (computationally challenging)
  - Complexity of decision rule tends to grow with # features. Hard to learn complex rules as it needs more data (statistically challenging)

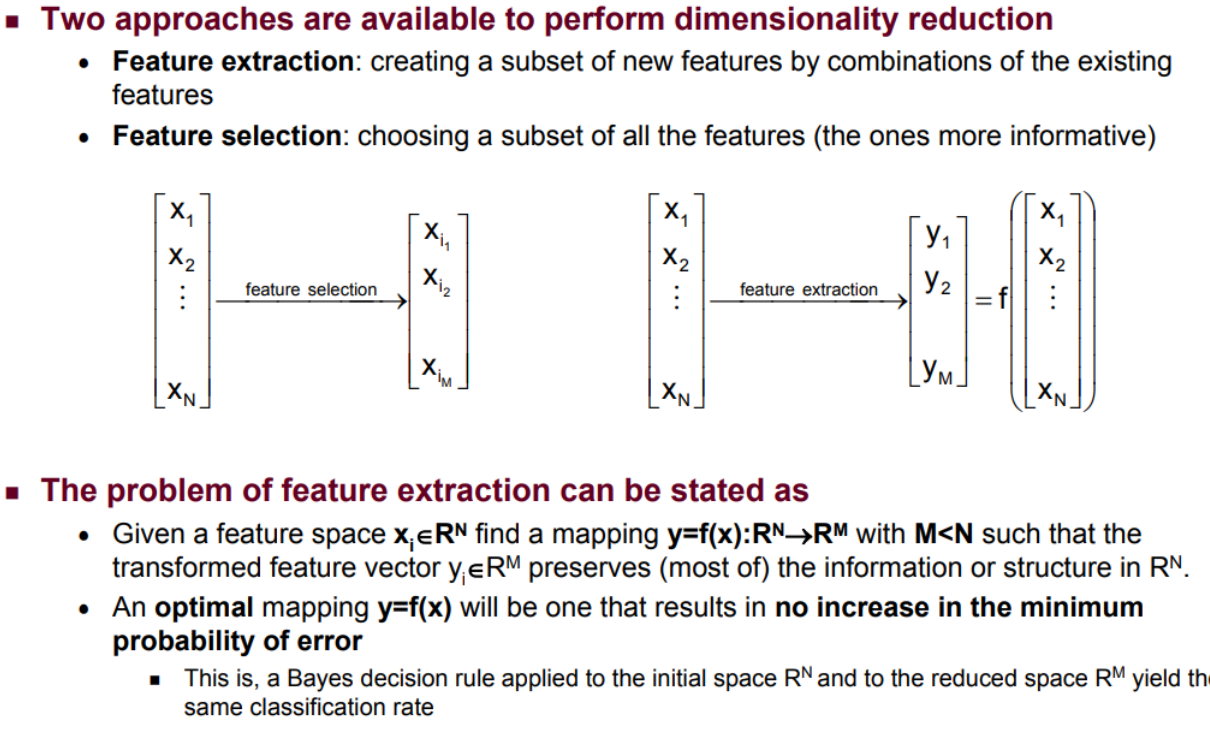
Addressing the Curse of Dimensionality

- Dimensionality Reduction: Techniques like Principal Component Analysis (PCA), t-SNE, and Autoencoders can reduce the number of dimensions while preserving the most critical information in the data.

- Feature Selection: Select a subset of the most informative features using methods like correlation analysis, mutual information, or tree-based feature importance.
- Regularization: Add constraints (regularization terms like L1 or L2) to models to prevent overfitting and handle high-dimensional data better.
- Distance Metric Learning: Learn a suitable distance metric for the high-dimensional space, ensuring that distances are meaningful.

There are two main approaches to dimensionality reduction:

- **Feature Selection:** In this approach, you select a subset of the original features and discard the rest. This subset contains the most relevant features for the task at hand. Common methods for feature selection include correlation analysis, mutual information, and filter methods.
- **Feature Extraction:** In this approach, you create new features that are combinations or transformations of the original features. Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are popular techniques for feature extraction. They find linear combinations of the original features that capture the most significant variation in the data.



Popular algorithms used for dimensionality reduction include principal component analysis (PCA).These algorithms seek to transform data from high-dimensional spaces to low-dimensional spaces without compromising meaningful properties in the original data. These techniques are typically deployed during exploratory data analysis (EDA) or data processing to prepare the data for modeling.

It's helpful to reduce the dimensionality of a dataset during EDA to help visualize data: this is because visualizing data in more than three dimensions is difficult. From a data processing perspective, reducing the dimensionality of the data simplifies the modeling problem.

16.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique in the fields of statistics, machine learning, and data analysis. It aims to transform high-dimensional data into a lower-dimensional representation while retaining the most important information and patterns.

PCA achieves this by finding a set of orthogonal axes, known as principal components, in the high-dimensional data space. Each principal component is a linear combination of the original features.

Here's a step-by-step explanation of how PCA works:

1. **Data Standardization:** Start with a dataset of high-dimensional data, where each column represents a feature or variable. Standardize the data by subtracting the mean and dividing by the standard deviation for each feature. This step ensures that all features have the same scale.

2. **Covariance Matrix Calculation:** Calculate the covariance matrix of the standardized data. The covariance matrix is a square matrix where each element represents the covariance between pairs of features.

3. **Eigenvalue Decomposition:** Perform eigenvalue decomposition on the covariance matrix to extract eigenvalues and eigenvectors. The eigenvalues represent the amount of variance explained by each principal component. The eigenvectors represent the directions in the high-dimensional space that maximize the variance.

4. **Sort Eigenvalues:** Order the eigenvalues in descending order. The first eigenvalue corresponds to the most significant variance, the second eigenvalue to the second-most significant variance, and so on.

5. **Select Principal Components:** Decide how many principal components you want to retain in the lower-dimensional representation. This choice can be based on the proportion of explained variance or specific requirements for dimensionality reduction.

6. **Projection:** Use the selected principal components (eigenvectors) to transform the data. Each data point is projected onto the subspace defined by these principal components. This transformation results in a lower-dimensional representation of the data.

7. **Variance Explained:** Calculate the proportion of total variance explained by the retained principal components. This information can help you assess the quality of the dimensionality reduction.

8. **Visualization and Analysis:** Visualize and analyze the lower-dimensional data to gain insights, identify patterns, or facilitate further data analysis. Principal components can be interpreted to understand the relationships between features in the original data.

9. **Inverse Transformation (Optional):** If necessary, you can perform an inverse transformation to map the reduced-dimensional data back to the original high-dimensional space. However, some information may be lost in this process.

10. **Application:** Use the lower-dimensional data for various tasks, such as visualization, clustering, classification, or regression, with reduced computational complexity and noise.

PCA provides several benefits:

- **Dimensionality Reduction:** By selecting a subset of principal components, you can reduce the dimensionality of your data while retaining most of the variance. This is especially useful when dealing with high-dimensional data.
- **Noise Reduction:** PCA can help filter out noise in the data, leading to cleaner and more interpretable patterns.
- **Visualization:** PCA facilitates the visualization of data in lower dimensions, making it easier to understand and explore complex datasets.
- **Feature Engineering:** PCA can serve as a form of feature engineering, identifying the most important features for a given task.

First watch the lectures then refer to below content

- **The objective of PCA is to perform dimensionality reduction while preserving as much of the randomness (variance) in the high-dimensional space as possible**
  - Let x be an N-dimensional random vector, represented as a linear combination of orthonormal basis vectors $[\varphi_1 | \varphi_2 | \ldots | \varphi_N]$ as

$$x = \sum_{i=1}^{N} y_i \varphi_i \quad \text{where} \quad \varphi_i \,|\, \varphi_j = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

  - Suppose we choose to represent x with only M (M<N) of the basis vectors. We can do this by replacing the components $[y_{M+1}, \ldots, y_N]^T$ with some pre-selected constants b_i

$$\hat{x}(M) = \sum_{i=1}^{M} y_i \varphi_i + \sum_{i=M+1}^{N} b_i \varphi_i$$

  - The representation error is then

$$\Delta x(M) = x - \hat{x}(M) = \sum_{i=1}^{N} y_i \varphi_i - \left(\sum_{i=1}^{M} y_i \varphi_i + \sum_{i=M+1}^{N} b_i \varphi_i \right) = \sum_{i=M+1}^{M} (y_i - b_i) \varphi_i$$

  - We can measure this representation error by the mean-squared magnitude of Δx
  - Our goal is to find the basis vectors φ_i and constants b_i that minimize this mean-square error

$$\bar{\varepsilon}^2(M) = E\left[|\Delta x(M)|^2 \right] = E\left[\sum_{i=M+1}^{N} \sum_{j=M+1}^{N} (y_i - b_i)(y_j - b_j) \varphi_i^T \varphi_j \right] = \sum_{i=M+1}^{N} E\left[(y_i - b_i)^2 \right]$$

- As we have done earlier in the course, the optimal values of b_i can be found by computing the partial derivative of the objective function and equating it to zero

$$\frac{\partial}{\partial b_i} E\left[(y_i - b_i)^2\right] = -2(E[y_i] - b_i) = 0 \Rightarrow b_i = E[y_i]$$

  - Therefore, we will replace the discarded dimensions y_i's by their expected value (an intuitive solution)
- The mean-square error can then be written as

$$\bar{\varepsilon}^2(M) = \sum_{i=M+1}^{N} E\left[(y_i - E[y_i])^2\right] = \sum_{i=M+1}^{N} E\left[(x\varphi_i - E[x\varphi_i])^T (x\varphi_i - E[x\varphi_i])\right]$$

$$= \sum_{i=M+1}^{N} \varphi_i^T E\left[(x - E[x])(x - E[x])^T\right]\varphi_i = \sum_{i=M+1}^{N} \varphi_i^T \Sigma_x \varphi_i$$

  - where Σ_x is the covariance matrix of x
- We seek to find the solution that minimizes this expression subject to the orthonormality constraint, which we incorporate into the expression using a set of Lagrange multipliers λ_i

$$\bar{\varepsilon}^2(M) = \sum_{i=M+1}^{N} \varphi_i^T \Sigma_x \varphi_i + \sum_{i=M+1}^{N} \lambda_i (1 - \varphi_i^T \varphi_i)$$

- Computing the partial derivative with respect to the basis vectors

$$\frac{\partial}{\partial \varphi_i} \bar{\varepsilon}^2(M) = \frac{\partial}{\partial \varphi_i}\left[\sum_{i=M+1}^{N} \varphi_i^T \Sigma_x \varphi_i + \sum_{i=M+1}^{N} \lambda_i (1 - \varphi_i^T \varphi_i)\right] = 2(\Sigma_x \varphi_i - \lambda_i \varphi_i) = 0 \Rightarrow \Sigma_x \varphi_i = \lambda_i \varphi_i$$

$$\text{NOTE: } \frac{d}{dx}(x^T A x) = (A + A^T)x \overset{\text{if A is symmmetric}}{=} 2Ax$$

  - So φ_i and λ_i are the eigenvectors and eigenvalues of the covariance matrix Σ_x

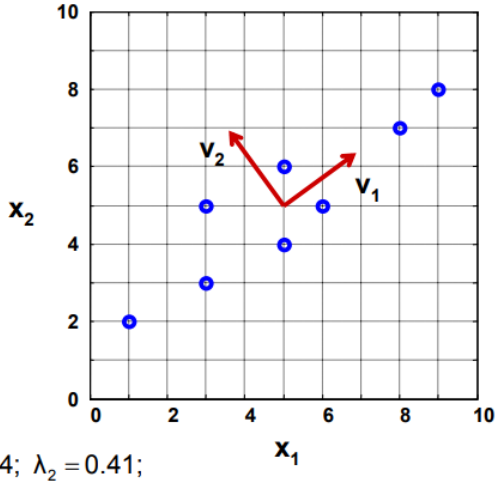
- We can then express the sum-square error as

$$\bar{\varepsilon}^2(M) = \sum_{i=M+1}^{N} \varphi_i^T \Sigma_x \varphi_i = \sum_{i=M+1}^{N} \varphi_i^T \lambda_i \varphi_i = \sum_{i=M+1}^{N} \lambda_i$$

- In order to minimize this measure, λ_i will have to be smallest eigenvalues
  - Therefore, to represent x with minimum sum-square error, we will choose the eigenvectors φ_i corresponding to the largest eigenvalues λ_i.

Example:

- **Compute the principal components for the following two-dimensional dataset**
  - $X=(x_1,x_2)=\{(1,2),(3,3),(3,5),(5,4),(5,6),(6,5),(8,7),(9,8)\}$
    - Let's first plot the data to get an idea of which solution we should expect



- **SOLUTION (by hand)**
  - The (biased) covariance estimate of the data is:

$$\Sigma_x = \begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix}$$

  - The eigenvalues are the zeros of the characteristic equation

$$\Sigma_x v = \lambda v \Rightarrow |\Sigma_x - \lambda I| = 0 \Rightarrow \begin{vmatrix} 6.25-\lambda & 4.25 \\ 4.25 & 3.5-\lambda \end{vmatrix} = 0 \Rightarrow \lambda_1 = 9.34;\ \lambda_2 = 0.41;$$

  - The eigenvectors are the solutions of the system

$$\begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix}\begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} = \begin{bmatrix} \lambda_1 v_{11} \\ \lambda_1 v_{12} \end{bmatrix} \Rightarrow \begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} = \begin{bmatrix} 0.81 \\ 0.59 \end{bmatrix}$$

$$\begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix}\begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix} = \begin{bmatrix} \lambda_2 v_{21} \\ \lambda_2 v_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix} = \begin{bmatrix} -0.59 \\ 0.81 \end{bmatrix}$$

    - HINT: To solve each system manually, first assume that one of the variables is equal to one (i.e. $v_{i1}=1$), then find the other one and finally normalize the vector to make it unit-length

- **NOTES**
  - Since PCA uses the eigenvectors of the covariance matrix Σ_x, it is able to find the independent axes of the data under the unimodal Gaussian assumption
    - For non-Gaussian or multi-modal Gaussian data, PCA simply de-correlates the axes
  - The main limitation of PCA is that it does not consider class separability since it does not take into account the class label of the feature vector
    - PCA simply performs a coordinate rotation that aligns the transformed axes with the directions of maximum variance
    - **There is no guarantee that the directions of maximum variance will contain good features for discrimination**
- **Historical remarks**
  - Principal Components Analysis is the oldest technique in multivariate analysis
  - PCA is also known as the Karhunen-Loève transform (communication theory)
  - PCA was first introduced by Pearson in 1901, and it experienced several modifications until it was generalized by Loève in 1963

16.2 Linear Discriminant Analysis (LDA)

- Linear Discriminant Analysis (LDA) is a fundamental technique in the fields of statistics, machine learning, and pattern recognition. It serves both as a classification method and a dimensionality reduction tool, enabling the differentiation of distinct classes within a dataset.
- Linear Discriminant Analysis (LDA) is a supervised learning algorithm primarily used for classification tasks and dimensionality reduction. Unlike unsupervised methods like Principal Component Analysis (PCA), LDA leverages class label information to maximize the separability between multiple classes.
- LDA was introduced by Ronald A. Fisher in 1936, primarily for distinguishing between species of Iris flowers. Fisher's work laid the groundwork for subsequent developments in statistical pattern recognition and machine learning.

Theoretical Framework

At its core, LDA seeks to find a linear combination of features that best separates two or more classes of objects or events. The key idea is to project high-dimensional data onto a lower-dimensional space while preserving class discriminatory information.

The objective is twofold:

- **Maximize Between-Class Variance:** Ensures that different classes are as far apart as possible.
- **Minimize Within-Class Variance:** Ensures that data points within the same class are as close to each other as possible.

Assumptions

LDA operates under several key assumptions:

- Normal Distribution: Features are assumed to follow a multivariate normal distribution within each class.
- Equal Covariance Matrices: All classes share the same covariance matrix, implying that they have similar shapes in feature space.
- Linearly Separable Classes: Classes can be separated by linear boundaries.Violations of these assumptions can affect the performance and reliability of LDA.

Violations of these assumptions can affect the performance and reliability of LDA.

Implementation Steps

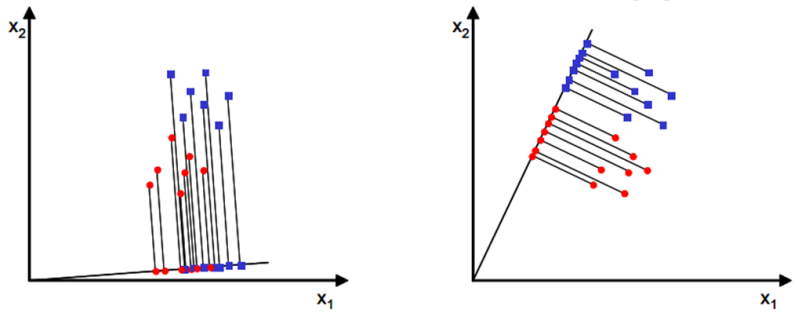
1. **Compute Mean Vectors:**
   - Calculate the mean vector for each class and the overall mean.
2. **Compute Scatter Matrices:**
   - Calculate $\mathbf{S}_W$ and $\mathbf{S}_B$.
3. **Solve the Generalized Eigenvalue Problem:**
   - Find eigenvectors and eigenvalues of $\mathbf{S}_W^{-1}\mathbf{S}_B$.
4. **Select Top Eigenvectors:**
   - Choose eigenvectors corresponding to the highest eigenvalues to form $\mathbf{W}$.
5. **Project Data:**
   - Transform the original data using $\mathbf{W}$ to obtain lower-dimensional representations.
6. **Classification:**
   - Use the projected data to build a classifier, often by applying simple methods like nearest centroid or linear classifiers.

Please watch the lecture for better understanding

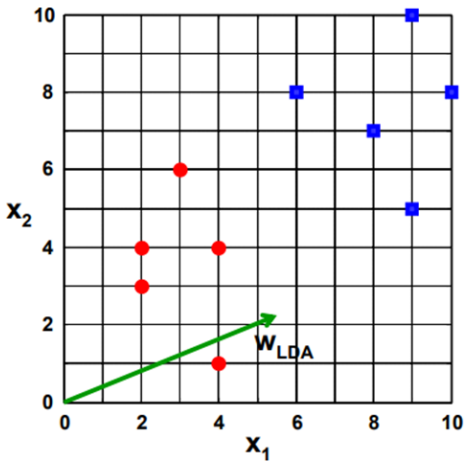
- **The objective of LDA is to perform dimensionality reduction while preserving as much of the class discriminatory information as possible**
  - Assume we have a set of D-dimensional samples $\{x^{(1}, x^{(2}, \ldots, x^{(N}\}$, N_1 of which belong to class ω_1, and N_2 to class ω_2. We seek to obtain a scalar y by projecting the samples x onto a line

$$y = w^T x$$

  - Of all the possible lines we would like to select the one that maximizes the separability of the scalars
    - This is illustrated for the two-dimensional case in the following figures



- **Compute the Linear Discriminant projection for the following two-dimensional dataset**
  - $X1=(x_1,x_2)=\{(4,1),(2,4),(2,3),(3,6),(4,4)\}$
  - $X2=(x_1,x_2)=\{(9,10),(6,8),(9,5),(8,7),(10,8)\}$



- **SOLUTION (by hand)**
  - The class statistics are:

$$S_1=\begin{bmatrix}0.80 & -0.40\\ -0.40 & 2.60\end{bmatrix};\ S_2=\begin{bmatrix}1.84 & -0.04\\ -0.04 & 2.64\end{bmatrix}$$

$$\mu_1=[3.00\quad 3.60];\qquad \mu_2=[8.40\quad 7.60]$$

  - The within- and between-class scatter are

$$S_B=\begin{bmatrix}29.16 & 21.60\\ 21.60 & 16.00\end{bmatrix};\ S_W=\begin{bmatrix}2.64 & -0.44\\ -0.44 & 5.28\end{bmatrix}$$

  - The LDA projection is then obtained as the solution of the generalized eigenvalue problem

$$S_W^{-1}S_B v=\lambda v \Rightarrow \left|S_W^{-1}S_B-\lambda I\right|=0 \Rightarrow \begin{vmatrix}11.89-\lambda & 8.81\\ 5.08 & 3.76-\lambda\end{vmatrix}=0 \Rightarrow \lambda=15.65$$

$$\begin{bmatrix}11.89 & 8.81\\ 5.08 & 3.76\end{bmatrix}\begin{bmatrix}v_1\\ v_2\end{bmatrix}=15.65\begin{bmatrix}v_1\\ v_2\end{bmatrix}\Rightarrow\begin{bmatrix}v_1\\ v_2\end{bmatrix}=\begin{bmatrix}0.91\\ 0.39\end{bmatrix}$$

  - Or directly by

$$w^*=S_W^{-1}(\mu_1-\mu_2)=[-0.91\quad -0.39]^T$$

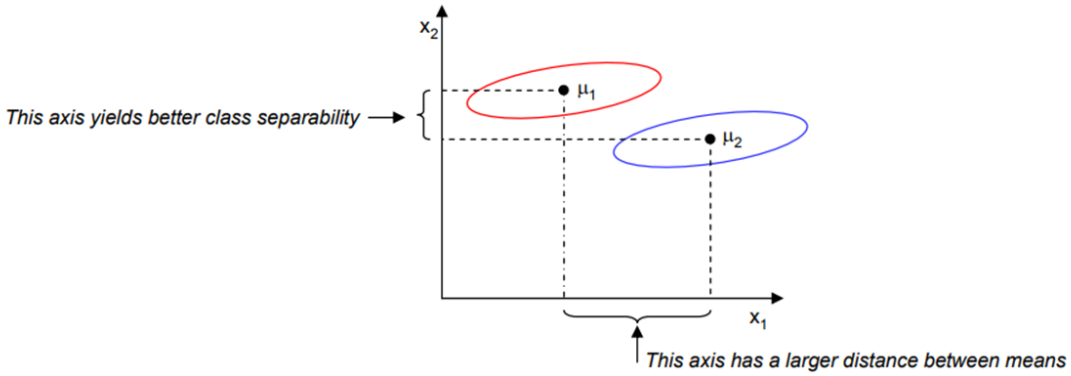
- **In order to find a good projection vector, we need to define a measure of separation between the projections**
  - The mean vector of each class in ***x*** and ***y*** feature space is

$$\mu_i=\frac{1}{N_i}\sum_{x\in\omega_i}x\quad\text{and}\quad\tilde{\mu}_i=\frac{1}{N_i}\sum_{y\in\omega_i}y=\frac{1}{N_i}\sum_{x\in\omega_i}w^Tx=w^T\mu_i$$

  - We could then choose the distance between the projected means as our objective function

$$J(w)=\left|\tilde{\mu}_1-\tilde{\mu}_2\right|=\left|w^T(\mu_1-\mu_2)\right|$$

  - However, the distance between the projected means is not a very good measure since it does not take into account the standard deviation within the classes



- **The solution proposed by Fisher is to maximize a function that represents the difference between the means, normalized by a measure of the within-class scatter**
  - For each class we define the scatter, an equivalent of the variance, as

$$\tilde{s}_i^2 = \sum_{y \in \omega_i} (y - \tilde{\mu}_i)^2$$

    - where the quantity $\left(\tilde{s}_1^2 + \tilde{s}_2^2\right)$ is called the within-class scatter of the projected examples
  - The Fisher linear discriminant is defined as the linear function $\mathbf{w^T x}$ that maximizes the criterion function

$$J(w) = \frac{\left|\tilde{\mu}_1 - \tilde{\mu}_2\right|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

- In order to find the optimum projection w*, we need to express J(w) as an explicit function of w
- We define a measure of the scatter in multivariate feature space **x**, which are scatter matrices

$$S_i = \sum_{x \in \omega_i} (x - \mu_i)(x - \mu_i)^T$$

$$S_1 + S_2 = S_W$$

  - where S_W is called the **within-class scatter matrix**
- The scatter of the projection **y** can then be expressed as a function of the scatter matrix in feature space **x**

$$\tilde{s}_i^2 = \sum_{y \in \omega_i} (y - \tilde{\mu}_i)^2 = \sum_{x \in \omega_i} \left(w^T x - w^T \mu_i\right)^2 = \sum_{x \in \omega_i} w^T (x - \mu_i)(x - \mu_i)^T w = w^T S_i w$$

$$\tilde{s}_1^2 + \tilde{s}_2^2 = w^T S_W w$$

- Similarly, the difference between the projected means can be expressed in terms of the means in the original feature space

$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = \left(w^T \mu_1 - w^T \mu_2\right)^2 = w^T \underbrace{(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T}_{S_B} w = w^T S_B w$$

  - The matrix S_B is called the **between-class scatter**. Note that, since S_B is the outer product of two vectors, its rank is at most one
- We can finally express the Fisher criterion in terms of S_W and S_B as

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

- To find the maximum of J(w) we derive and equate to zero

$$\frac{d}{dw}[J(w)] = \frac{d}{dw}\left[\frac{w^T S_B w}{w^T S_W w}\right] = 0 \Rightarrow$$
$$\Rightarrow [w^T S_W w]\frac{d[w^T S_B w]}{dw} - [w^T S_B w]\frac{d[w^T S_W w]}{dw} = 0 \Rightarrow$$
$$\Rightarrow [w^T S_W w]2S_B w - [w^T S_B w]2S_W w = 0$$

- Dividing by $w^T S_W w$

$$\frac{[w^T S_W w]}{[w^T S_W w]}S_B w - \frac{[w^T S_B w]}{[w^T S_W w]}S_W w = 0 \Rightarrow$$
$$\Rightarrow S_B w - J S_W w = 0 \Rightarrow$$
$$\Rightarrow S_W^{-1} S_B w - Jw = 0$$

- Solving the generalized eigenvalue problem ($S_W^{-1} S_B w = Jw$) yields

$$w^* = \underset{w}{\text{argmax}}\left\{\frac{w^T S_B w}{w^T S_W w}\right\} = S_W^{-1}(\mu_1 - \mu_2)$$

  - This is know as **Fisher's Linear Discriminant** (1936), although it is not a discriminant but rather a specific choice of direction for the projection of the data down to one dimension

17 Performance Metrics

Performance metrics are used to evaluate how well a model performs, typically by comparing its predictions against actual outcomes. The choice of metric depends on the problem type (e.g., classification, regression) and the goals of the model.

17.1 Confusion Matrix:

A confusion matrix is a table used in machine learning to evaluate the performance of a classification model, particularly in supervised learning. It provides a detailed breakdown of the model's correct and incorrect predictions, allowing you to understand not just how often the model is right or wrong, but the types of errors it makes.

For a binary classification problem, the confusion matrix is a 2x2 table that compares the actual labels with the predicted labels:

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Definitions:

- **True Positives (TP):** Cases where the model correctly predicts the positive class.
- **True Negatives (TN):** Cases where the model correctly predicts the negative class.
- **False Positives (FP):** Cases where the model incorrectly predicts the positive class (Type I error).
- **False Negatives (FN):** Cases where the model incorrectly predicts the negative class (Type II error).

- **Precision**: The proportion of positive identifications that were actually correct.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall (Sensitivity)**: The proportion of actual positives that were correctly identified.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1 Score**: The harmonic mean of precision and recall.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Specificity**: The proportion of actual negatives that were correctly identified.

$$\text{Specificity} = \frac{TN}{TN + FP}$$

- **Accuracy**: The proportion of total correct predictions.

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

AUC and ROC

AUC (Area Under the Curve) and ROC (Receiver Operating Characteristic) are important metrics in evaluating the performance of binary classification models in machine learning.
The ROC curve plots the true positive rate (recall) versus the false positive rate. AUC measures the overall performance across all thresholds.

The ROC curve is a graphical representation of a classifier's performance across all classification thresholds. It plots two metrics:

- **True Positive Rate (TPR)**: Also known as Sensitivity or Recall, it measures the proportion of actual positives that are correctly identified.

$$\text{TPR} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **False Positive Rate (FPR)**: It measures the proportion of actual negatives that are incorrectly identified as positives.

$$\text{FPR} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

The ROC curve is created by plotting the TPR against the FPR at various threshold levels.

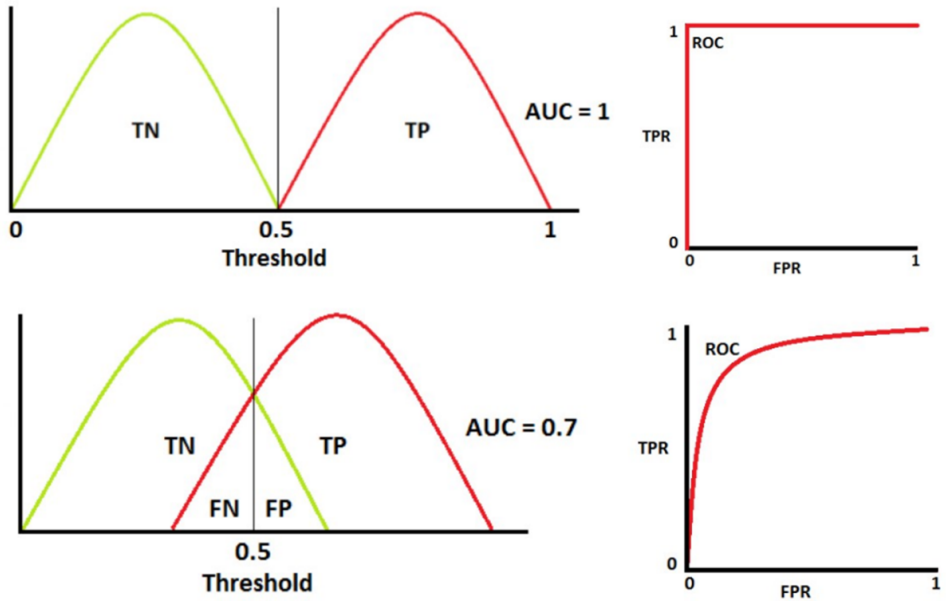
AUC (Area Under the ROC Curve)

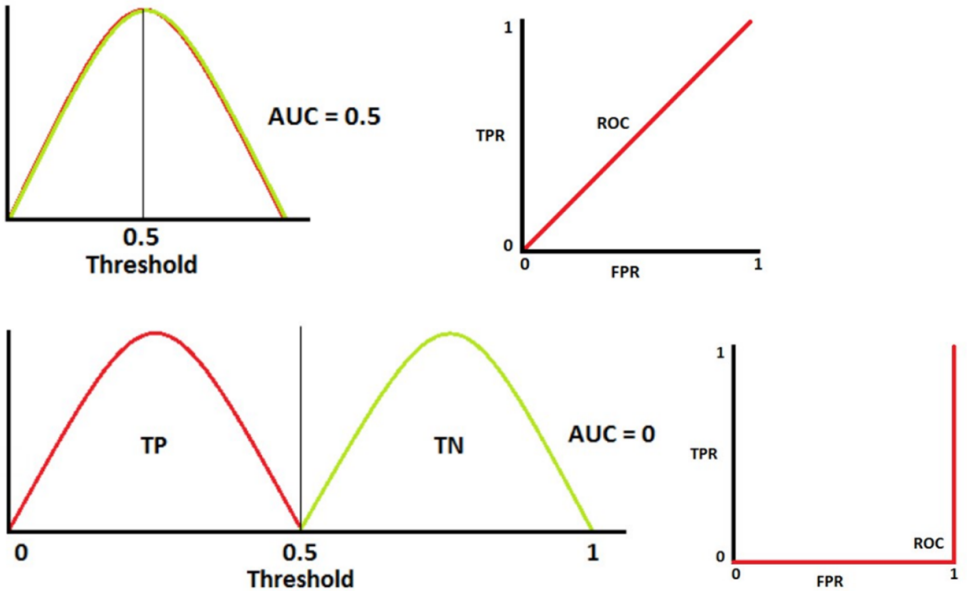
AUC quantifies the overall performance of a model by calculating the area under the ROC curve. It provides a single value that summarizes the model's ability to discriminate between the positive and negative classes.

- **AUC = 1**: Perfect model (no false positives or false negatives).
- **AUC = 0.5**: Model performs no better than random chance.
- **AUC < 0.5**: Indicates a model that is performing worse than random guessing.

Parameters and ROC

Please watch the lectures for better understanding





17.2 Regression Metrics:

These metrics are used when the task involves predicting continuous values.

- **Mean Absolute Error (MAE)**: The average of the absolute differences between predicted and actual values. It provides a clear indication of how far off predictions are.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- **Mean Squared Error (MSE)**: The average of the squared differences between predicted and actual values. It gives more weight to larger errors.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- **Root Mean Squared Error (RMSE)**: The square root of MSE, which brings the error back to the original scale of the target variable.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- **R-squared (R^2)**: The proportion of the variance in the target variable that is predictable from the features. A value closer to 1 indicates a better fit.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

- **n**: The total number of data points (samples) in the dataset.
- y_i: The actual value of the target variable for the i^{th} data point. This is the ground truth value.
- $\hat{y}_i$: The predicted value of the target variable for the i^{th} data point. This is the model's prediction.

Choosing the Right Metric:

The selection of a performance metric should be aligned with the problem's goals:

- For imbalanced classes, accuracy can be misleading, and metrics like precision, recall, and F1-score are often more informative.
- In regression tasks, MAE, MSE, and R^2 are commonly used, but the choice depends on how sensitive the model should be to large errors.
- For clustering, a combination of internal and external validation metrics helps to evaluate the quality of the clustering.

References

Lecture Notes in MACHINE LEARNING, by Dr V N Krishnachandran

https://alexjungaalto.github.io/MLBasicsBook.pdf

Taeho Jo Machine Learning Foundations Supervised, Unsupervised, and Advanced Learning Springer book

IIT Madras BS Degree Lectures and Notes

NPTEL Lectures and Slides

www.medium.com

geeksforgeeks.org/

javatpoint.com/