TRIBHUVAN UNIVERSITY

FACULTY OF HUMANITIES AND SOCIAL SCIENCES

BACHELOR OF COMPUTER APPLICATION (BCA)

BCA	CACS456 MACHINE LEARNING (ML)

Contents

UNIT 1 INTRODUCTION TO MACHINE LEARNING....1
History of Machine Learning....1
Introduction to Machine Learning....2
Brain – neuron learning system....7
Definition and Types of learning models....7
Need of Machine Learning....18
Data and tools....19
Review of statistics....19
Training, Validation and Test data....20
Theory of learning....21
Generalization Bound:....21
Approximation-generalization tradeoff....22
Bias and variance....22
Learning curve....23
UNIT 2 SUPERVISED LEARNING....24
Classification Problems....24
Regression:....25
Decision Tree Representation:....31
Appropriate Problem for Decision Tree Learning:....31
Basic Decision tree algorithm:....32
Support Vector Machine:....37
Maximal margin hyperplanes....39
Finding maximum margin:....40
UNIT 3 BAYESIAN AND INSTANCE BASED LEARNING....41
Bayes' Rule (Theorem)....42
Maximum likelihood estimation (ML estimation)....46
The general MLE method....47
Bayesian Belief networks....47
K-Nearest Neighbor(KNN) Algorithm....49
UNIT 4 INTRODUCTION TO UN-SUPERVISED LEARNING AND DIMENSIONALITY REDUCTION....53
Introduction to Clustering....53
K-means clustering....53
Hierarchical clustering....58
Principal component analysis....60
UNIT 5 MEASURES FOR PERFORMANCE EVALUATION OF ML ALGORITHMS....64
Classification Accuracy....64
Confusion matrix....65
Precision and recall....65
Misclassification costs....66
Sensitivity and specificity....67
Receiver Operating Characteristic (ROC)....68

UNIT 1

INTRODUCTION TO MACHINE LEARNING

History of Machine Learning

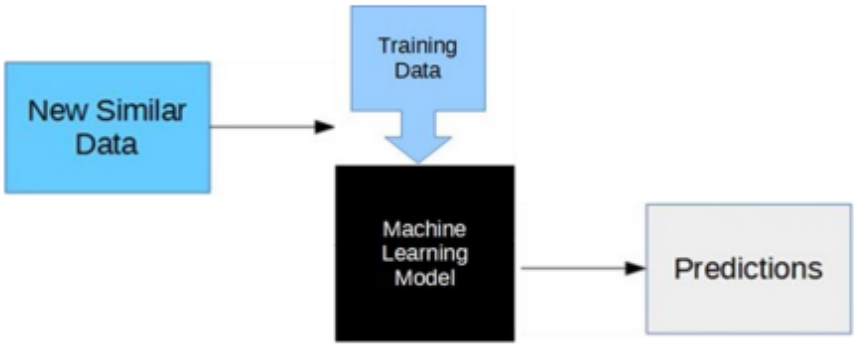
- some major milestones and developments in the history of machine learning:
  - 1943: Warren McCulloch and Walter Pitts develop the first mathematical model of an artificial neuron.
  - 1949: Donald Hebb proposes the Hebbian learning rule, which describes how neurons in the brain can learn and adapt based on their interactions with other neurons.
  - 1950: The Turing Test: Alan Turing proposed the Turing Test-according to which, if a machine can convince a human being into believing that the machine is also a human, then it is deemed "intelligent".
  - 1952: Arthur Samuel develops the first machine learning program, a checkers-playing program that improves its performance over time through self-play and reinforcement learning.
  - 1957: Frank Rosenblatt invents the perceptron, a type of artificial neural network that can learn to recognize patterns in data.
  - 1967: The nearest neighbor algorithm is developed, which uses a database of labeled examples to classify new examples based on their similarity to the known examples.
  - 1979: Tom Mitchell defines machine learning as the "study of algorithms that improve their performance at some task with experience."
  - 1981: Gerald Dejong introduces the concept of explanation-based learning, in which a machine learning algorithm learns from specific examples and generalizes its knowledge to new situations.
  - 1985: The backpropagation algorithm is developed, which allows artificial neural networks with multiple layers to learn from data.
  - 1995: The support vector machine (SVM) algorithm is developed, which uses a hyperplane to separate data into different classes and has applications in image and speech recognition.
  - 1997: The first International Conference on Machine Learning is held, marking the beginning of a period of rapid growth and innovation in the field.
  - 2006: Geoffrey Hinton introduces deep learning, a type of artificial neural network with multiple layers that can learn to recognize complex patterns in data.
  - 2012: Alex Krizhevsky develops a deep learning algorithm called AlexNet that achieves breakthrough results in image classification and helps to popularize deep learning.
  - 2015: Google's AlphaGo program defeats a world champion at the game of Go, demonstrating the power of deep reinforcement learning and spurring interest in the field.
  - 2018: OpenAI develops GPT-2, a language model based on deep learning that can generate human-like text.
  - 2020: GPT-3, a larger and more powerful version of GPT-2, is released and receives widespread attention for its ability to perform a wide range of natural language tasks.

Introduction to Machine Learning

Arthur Samuel, an early American leader in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. He defined machine learning as "the field of study that gives computers the ability to learn without being explicitly programmed." However, there is no universally accepted definition for machine learning.

Different authors define the term differently. We give below two more definitions:

1. 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
2. The field of study known as machine learning is concerned with the question of how to construct computer programs that automatically improve with experience.



Note: In the above definitions we have used the term "model" and we will be using this term at several contexts. It appears that there is no universally accepted one sentence definition of this term. Loosely, it may be understood as some mathematical expression or equation, or some mathematical structures such as graphs and trees, or a division of sets into disjoint subsets, or a set of logical "if . . . then . . . else . . ." rules, or some such thing. It may be noted that this is not an exhaustive list.

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.

Examples

i) **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

ii) **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

iii) **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

"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".

Basic components of learning process

The learning process, whether by a human or a machine, can be divided into four components, namely, data storage, abstraction, generalization and evaluation. Figure below illustrates the various components and the steps involved in the learning process

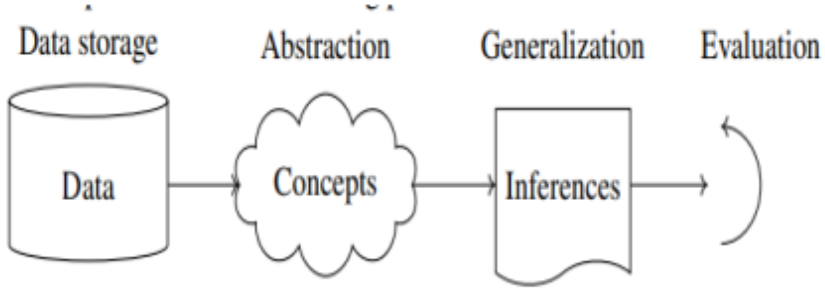


Fig 1: Components of learning Process

1. Data storage

Facilities for storing and retrieving huge amounts of data are an important component of the learning process. Humans and computers alike utilize data storage as a foundation for advanced reasoning.

- In a human being, the data is stored in the brain and data is retrieved using electrochemical signals.
- Computers use hard disk drives, flash memory, random access memory and similar devices to store data and use cables and other technology to retrieve data.

2. Abstraction

The second component of the learning process is known as abstraction. Abstraction is the process of extracting knowledge about stored data. This involves creating general concepts about the data as a whole. The creation of knowledge involves application of known models and creation of new models. The process of fitting a model to a dataset is known as training. When the model has been trained, the data is transformed into an abstract form that summarizes the original information.

3. Generalization

The third component of the learning process is known as generalisation. The term generalization describes the process of turning the knowledge about stored data into a form that can be utilized for future action. These actions are to be carried out on tasks that are similar, but not identical, to those what have been seen before. In generalization, the goal is to discover those properties of the data that will be most relevant to future tasks.

4. Evaluation

Evaluation is the last component of the learning process. It is the process of giving feedback to the user to measure the utility of the learned knowledge. This feedback is then utilised to effect improvements in the whole learning process.

Applications of machine learning

Application of machine learning methods to large databases is called data mining. In data mining, a large volume of data is processed to construct a simple model with valuable use, for example, having high predictive accuracy.

The following is a list of some of the typical applications of machine learning.

- In retail business, machine learning is used to study consumer behaviour.
- In finance, banks analyze their past data to build models to use in credit applications, fraud detection, and the stock market.
- In manufacturing, learning models are used for optimization, control, and troubleshooting.
- In medicine, learning programs are used for medical diagnosis.
- In telecommunications, call patterns are analyzed for network optimization and maximizing the quality of service.
- In science, large amounts of data in physics, astronomy, and biology can only be analyzed fast enough by computers. The World Wide Web is huge; it is constantly growing and searching for relevant information cannot be done manually.
- In artificial intelligence, it is used to teach a system to learn and adapt to changes so that the system designer need not foresee and provide solutions for all possible situations.
- It is used to find solutions to many problems in vision, speech recognition, and robotics.
- Machine learning methods are applied in the design of computer-controlled vehicles to steer correctly when driving on a variety of roads.
- Machine learning methods have been used to develop programmes for playing games such as chess, backgammon and GO

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.

3. Spam e-mail

Let it be required to build a learning algorithm to identify spam e-mail.

(a) The unit of observation could be an e-mail messages.

(b) The examples would be specific messages.

(c) The features might consist of the words used in the messages.

Note: Examples and features are generally collected in a "matrix format".

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.1.2, the features "year", "price" and "mileage" are numeric and the features "model", "color" and "transmission" are categorical

features

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

examples

Fig 2: Example for "examples" and "features" collected in a matrix format (data relates to automobiles and their features)

Framework for building a machine learning system

When building a machine learning system, you typically need to follow a structured framework to ensure a systematic approach and achieve optimal results. Here is a general framework that you can use as a guide:

1. ***Define the Problem***: Clearly define the problem you want to solve with machine learning. This involves understanding the objectives, requirements, and constraints of the problem.
2. ***Gather Data***: Collect the relevant data needed to solve the problem. This could involve acquiring data from various sources, such as databases, APIs, or data scraping. Ensure that the data is representative, of good quality, and suitable for the task at hand.
3. ***Data Preprocessing***: Clean and preprocess the data to prepare it for the machine learning algorithms. This step may involve handling missing values, dealing with outliers, normalizing or scaling features, and performing feature engineering to create meaningful representations.
4. ***Split the Data:*** Split the dataset into training, validation, and testing sets. The training set is used to train the models, the validation set is used for hyper parameter tuning and model selection, and the testing set is used to evaluate the final model's performance.
5. ***Select a Model:*** Choose an appropriate machine learning algorithm or model that is suitable for your problem. This could include various types of models such as linear regression, decision trees, support vector machines, neural networks, or ensemble methods.
6. ***Train the Model***: Train the selected model using the training data. This involves feeding the data into the model, optimizing its parameters, and adjusting the model's internal settings to minimize the error or maximize performance on the training set.

7. ***Evaluate the Model***: Assess the model's performance using the validation set. This involves evaluating various metrics such as accuracy, precision, recall, F1-score, or mean squared error, depending on the problem type (classification, regression, etc.).
8. ***Hyper parameter Tuning***: Fine-tune the model by adjusting its hyper parameters to improve its performance. This can be done through techniques like grid search, random search, or more advanced methods like Bayesian optimization or genetic algorithms.
9. ***Test the Model***: Once you are satisfied with the model's performance on the validation set, evaluate it on the independent testing set to get a final estimate of its performance. This step helps you assess how well your model generalizes to unseen data.
10. ***Deploy and Monitor***: If the model performs well, deploy it to a production environment and monitor its performance in real-world scenarios. Continuously monitor and evaluate the model's predictions, retraining or updating it as necessary to maintain optimal performance.

It's important to note that this is a high-level framework, and the specific steps and techniques involved may vary depending on the problem domain, data availability, and the algorithms being used. Additionally, it's crucial to follow ethical guidelines and consider factors such as bias, fairness, and interpretability throughout the entire process.

Brain – neuron learning system

- The human brain is the inspiration for developing artificial neural networks.
- Biological neurons are cells that process and transmit information in the brain and nervous system.
- In artificial neural networks, each neuron is represented by a node that takes in input signals from other nodes through weighted connections.
- The input signals are transformed by an activation function to determine whether the neuron should fire an output signal.
- Learning in both biological and artificial neural networks occurs through the strengthening or weakening of the connections between neurons.
- In biological neurons, this is achieved through synaptic plasticity.
- In artificial neural networks, this is achieved through training algorithms such as back propagation.
- The brain-neuron learning system is a complex and dynamic process that involves many different factors and mechanisms.

Definition and Types of learning models

- Learning refers to the process of acquiring knowledge, skills, or behaviors through experience, study, or instruction. It is a fundamental aspect of human and animal cognition and is also a key area of study in machine learning and artificial intelligence.

The different types of learning models mentioned

Learning Models:

Machine learning models are algorithms or mathematical representations that are trained on data to make predictions or perform specific tasks. Here are some commonly used machine learning models:

1. ***Linear Regression***: A model used for regression tasks, where the goal is to predict a continuous numerical value based on input features. It assumes a linear relationship between the features and the target variable.
2. ***Logistic Regression***: A model used for binary classification tasks, where the goal is to predict one of two classes. It estimates the probability of the input belonging to a certain class using a logistic function.
3. ***Decision Trees:*** A model that uses a tree-like structure to make decisions based on feature values. Each internal node represents a decision based on a feature, and each leaf node represents a predicted class or value.
4. ***Random Forest:*** An ensemble model that combines multiple decision trees to make predictions. It aggregates the predictions of individual trees to produce a final prediction.
5. ***Support Vector Machines (SVM):*** A model that separates data points into different classes using hyperplanes in high-dimensional spaces. It aims to maximize the margin between classes.
6. ***Neural Networks***: Deep learning models composed of interconnected layers of artificial neurons. They can learn complex patterns and relationships from data, making them suitable for tasks such as image recognition and natural language processing.
7. ***Naive Bayes***: A probabilistic model based on Bayes' theorem that assumes independence among features. It is commonly used for text classification tasks such as spam detection and sentiment analysis.
8. ***K-Nearest Neighbors (KNN):*** A model that classifies data points based on the classes of their nearest neighbors. It assigns a new point to the majority class among its k nearest neighbors.
9. ***Gradient Boosting Models:*** Ensemble models that combine multiple weak learners (typically decision trees) to create a strong learner. Examples include Gradient Boosting Machines (GBM), XGBoost, and LightGBM.
10. ***Hidden Markov Models (HMM):*** A statistical model used for sequence data, where the underlying system is assumed to be a Markov process with hidden states. HMMs are widely used in speech recognition and bioinformatics.

These are just a few examples of machine learning models, and there are many more algorithms and variations available depending on the specific problem and data characteristics. Each model has its strengths, limitations, and suitability for different types of tasks.

Geometric models

Geometric models refer to mathematical representations or structures that capture the spatial relationships and properties of objects or systems. These models are used in various fields, including computer graphics, computer-aided design (CAD), computer vision, robotics, and physics. Here are a few examples of geometric models:

1. ***Point Clouds:*** A point cloud is a collection of data points in three-dimensional space. It represents the shape or surface of an object by capturing the spatial coordinates of individual points. Point clouds are often used in 3D scanning, object recognition, and reconstruction.
2. ***Mesh Models:*** Mesh models are composed of vertices, edges, and faces that define the shape and surface of an object. Triangular meshes, where each face is a triangle, are commonly used in computer graphics and visualization. Mesh models are used for 3D rendering, simulation, and animation.

3. ***Bézier Curves and Surfaces:*** Bézier curves are mathematical curves defined by control points. They are widely used in computer graphics and CAD for representing smooth curves. Bézier surfaces extend this concept to represent smoothly varying surfaces.
4. ***Implicit Models***: Implicit models represent objects or surfaces as the zero level set of a continuous function. For example, the implicit representation of a sphere is defined by the equation of a sphere. Implicit models are used in shape modeling, collision detection, and image segmentation.
5. ***Octrees:*** An octree is a tree-based data structure used for representing 3D space. It recursively subdivides space into eight equal-sized octants. Octrees are efficient for spatial indexing, collision detection, and efficient representation of volumetric data.
6. ***Solid Models:*** Solid models represent the interior and exterior volumes of objects.

Probabilistic models:

Probabilistic models are a class of models that use probability theory to represent uncertainty and make predictions or decisions based on probabilistic reasoning. These models capture the inherent uncertainty in data and provide a framework for reasoning about uncertain events or variables. Here are a few examples of probabilistic models:

1. ***Naive Bayes***: Naive Bayes is a simple probabilistic model used for classification tasks. It assumes that the features are conditionally independent given the class label. The model calculates the probability of a class given the observed features using Bayes' theorem.
2. ***Hidden Markov Models (HMMs):*** HMMs are probabilistic models widely used in sequential data analysis. They consist of a set of hidden states and observable outputs. The transitions between hidden states are governed by probabilities, and the model can be used to infer the hidden states given the observed sequence (e.g., speech recognition, part-of-speech tagging).
3. ***Gaussian Mixture Models (GMMs):*** GMMs are probabilistic models used for density estimation and clustering tasks. They represent the data distribution as a mixture of Gaussian components, where each component is associated with a probability. GMMs are often used for modeling complex data distributions and for applications like image segmentation or anomaly detection.
4. ***Bayesian Networks***: Bayesian networks are graphical models that represent probabilistic relationships among a set of variables. They use directed acyclic graphs to capture the dependencies between variables and conditional probability tables to represent the probabilistic relationships. Bayesian networks are widely used in decision making, risk analysis, and causal reasoning.
5. ***Probabilistic Graphical Models (PGMs):*** PGMs are a general framework that combines graph theory and probability theory to model complex dependencies and uncertainties. They include Bayesian networks, Markov networks (also known as Markov random fields), and their combinations. PGMs are used for tasks like reasoning, inference, and learning from uncertain data.
6. ***Variational Autoencoders (VAEs):*** VAEs are generative models that use probabilistic inference to learn a low-dimensional representation (latent space) of high-dimensional data. They combine deep neural networks with variational inference to model the probability distribution of the data and generate new samples.

7. ***Probabilistic Programming:*** Probabilistic programming languages (PPLs) provide a higher-level programming interface to define and reason about probabilistic models. They allow the specification of complex models using probabilistic primitives and provide inference algorithms to perform probabilistic reasoning.

Logic Models:

Logic models, also known as program logic models or program theory models, are tools used in program evaluation and planning to visually represent the logical relationships between program inputs, activities, outputs, outcomes, and impacts. They provide a structured framework for understanding how a program is expected to work and the expected outcomes it aims to achieve. Here are the main components of a logic model:

1. **Inputs**: Inputs refer to the resources, such as funding, staff, and materials, that are allocated to the program.
2. **Activities**: Activities represent the specific actions or interventions undertaken by the program to address the problem or achieve the desired outcomes. These can include training sessions, workshops, counseling, or any other program-related activities.
3. **Outputs**: Outputs are the direct products or deliverables of the program activities. They are the immediate results of the program's efforts and can be measured quantitatively or qualitatively. Examples of outputs include the number of participants trained, the number of counseling sessions conducted, or the number of educational materials distributed.
4. **Outcomes**: Outcomes are the changes or effects that occur as a result of the program's activities and outputs. They represent the short-term, intermediate, or long-term changes that the program aims to achieve. Outcomes can be divided into three categories:
    a. ***Short-term Outcomes****: Immediate changes that occur shortly after program activities and outputs, such as increased knowledge or improved skills.*
    b. ***Intermediate Outcomes****: Medium-term changes that occur as a result of sustained program engagement and interaction. These outcomes often involve changes in behavior, attitudes, or practices.*
    c. ***Long-term Outcomes****: The ultimate goals or impacts of the program, which may take a longer time to achieve. These outcomes are often related to broader social, economic, or health indicators.*
5. **Impact**: The highest level of change or impact that the program seeks to achieve. It represents the long-term societal, environmental, or systemic changes resulting from the program's outcomes. Impacts are typically aligned with the program's mission and long-term goals.

Logic models help stakeholders understand the underlying theory of how a program is expected to work, identify key evaluation questions, and guide the selection of appropriate indicators and evaluation methods. They provide a visual representation of the logical relationships between program components and aid in communicating program design and expected outcomes to various stakeholders.

Grouping and grading:

Grouping and grading are two practices commonly used in various contexts to organize and categorize entities or individuals based on specific criteria. Here's an overview of each concept:

Grouping:

Grouping refers to the process of categorizing or classifying entities or objects into distinct groups based on shared characteristics, properties, or criteria. The purpose of grouping is to organize similar items together, which facilitates analysis, understanding, and decision-making. Grouping can be applied in various domains, such as education, market segmentation, data analysis, and social sciences.

Examples of grouping include:

1. Students in a classroom being grouped based on their academic performance or learning styles.
2. Customers being segmented into different groups based on their demographics, behaviors, or preferences for targeted marketing campaigns.
3. Data points in clustering algorithms being grouped into clusters based on similarity or proximity in feature space.
4. Participants in a research study being grouped into experimental and control groups for comparative analysis.

Grading:

Grading, on the other hand, involves assigning grades or ratings to individuals or entities based on predefined criteria or standards. It is commonly used in educational settings to evaluate student performance, but it can also be employed in various contexts where assessment or evaluation is necessary.

Examples of grading include:

1. Teachers assigning grades (e.g., A, B, C) to students' assignments, tests, or overall academic performance.
2. Evaluators providing ratings or scores to participants in a competition, such as sports, art, or talent shows.
3. Performance appraisals in workplaces where employees are graded based on their achievements, skills, or job performance.
4. Quality assessments in product or service reviews, where ratings are given based on specific criteria such as usability, durability, or customer satisfaction.

Grading allows for the differentiation and ranking of individuals or entities based on predefined standards or criteria, providing a means to compare performance or quality. It can serve as a basis for making decisions, providing feedback, or recognizing excellence. However, it's important to consider the limitations and potential biases that grading systems may introduce and ensure fairness and transparency in the process.

In the context of machine learning, grouping and grading can be applied to various tasks and techniques.

Here are some ways in which grouping and grading are relevant in machine learning:

Grouping:

1. **Clustering**: Clustering algorithms group similar data points together based on their features or characteristics. It helps in identifying natural clusters or patterns within datasets without predefined class labels. Common clustering algorithms include K-means, hierarchical clustering, and DBSCAN.

2. **Market Segmentation**: In machine learning applications for marketing, customer segmentation involves grouping individuals with similar behaviors, preferences, or characteristics. It helps businesses tailor marketing strategies and target specific customer segments with personalized offers or recommendations.
3. **Anomaly Detection**: Grouping is also used for identifying anomalous or outlier data points within a dataset. Anomaly detection algorithms aim to find instances that deviate significantly from the expected patterns or distributions. It is useful in detecting fraudulent transactions, network intrusions, or unusual behavior in sensor data.

Grading:

1. **Regression**: Regression models estimate continuous values or scores based on input features. They can be used for grading or rating purposes, such as predicting the price of a house based on its features or assigning a numerical score to a product based on its characteristics.
2. **Sentiment Analysis**: Sentiment analysis algorithms classify text or reviews into positive, negative, or neutral sentiment. They can be used to grade or assess the sentiment or opinion of users towards products, services, or events.
3. **Recommendation Systems**: Recommendation algorithms grade or rank items based on user preferences or behavior. They predict the likelihood of a user's interest in an item and provide personalized recommendations. Grading helps prioritize or rank items for recommendation based on user preferences.
4. **Quality Assessment**: Machine learning models can be trained to grade or assess the quality of products, images, or audio based on specific criteria or standards. For example, an image quality assessment model can assign a score to images based on metrics like sharpness, noise, or color accuracy.

Designing a learning system:

Designing a learning system involves several key steps and considerations to ensure its effectiveness and success. Here is a general framework for designing a learning system:

1. **Identify Learning Goals**: Start by defining clear learning goals and objectives. Determine what knowledge, skills, or competencies the learners should acquire or develop through the system. This step helps provide a clear direction for the design process.
2. **Analyze the Target Audience**: Understand the characteristics, needs, and preferences of the target audience. Consider their prior knowledge, learning styles, motivations, and any specific requirements or constraints they may have. This analysis will guide the selection of appropriate instructional strategies and content delivery methods.
3. **Define Learning Content**: Determine the content that needs to be covered to achieve the learning goals. Break down the content into modules, topics, or learning units. Organize the content in a logical sequence and ensure it aligns with the identified learning goals.
4. **Choose Instructional Strategies**: Select appropriate instructional strategies and methods to deliver the content effectively. Consider a mix of instructional techniques, such as lectures, discussions, interactive activities, simulations, or hands-on exercises. Choose strategies that engage learners, promote active learning, and cater to different learning styles.

5. **Develop Learning Materials**: Create or curate learning materials that support the instructional strategies and content delivery. This may include text-based materials, multimedia resources, interactive modules, quizzes, assessments, or supplementary materials. Ensure the materials are engaging, accessible, and align with the identified learning goals.
6. **Incorporate Assessments**: Design assessments and evaluation methods to measure learners' progress and achievement of learning goals. Include formative assessments throughout the learning process to provide feedback and guide learners' understanding. Consider summative assessments at the end of modules or courses to evaluate overall learning outcomes.
7. **Consider Learning Technologies**: Determine the appropriate learning technologies or platforms to deliver the learning system. This could involve learning management systems (LMS), online platforms, virtual classrooms, or mobile applications. Select technologies that support the instructional strategies, content delivery, and learner interactions effectively.
8. **Ensure Interactivity and Engagement**: Design interactive elements and activities that promote learner engagement and participation. Incorporate opportunities for learners to apply knowledge, solve problems, collaborate with peers, or engage in discussions. Use multimedia elements, simulations, case studies, or real-world examples to enhance interactivity.
9. **Provide Support and Resources**: Consider the support mechanisms and resources needed to assist learners throughout their learning journey. This may involve providing access to tutors, mentors, or subject matter experts. Offer supplementary resources, references, or online communities for additional support and deeper exploration of the topics.
10. **Evaluate and Improve**: Implement an ongoing evaluation process to assess the effectiveness of the learning system. Collect feedback from learners and stakeholders to identify strengths, weaknesses, and areas for improvement. Use this feedback to refine the design, content, and delivery methods of the learning system.

Note: Design of a learning system should be iterative and adaptable. Continuously monitor the system's performance, gather feedback, and make necessary adjustments to ensure optimal learning outcomes.

Different types of learning

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

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

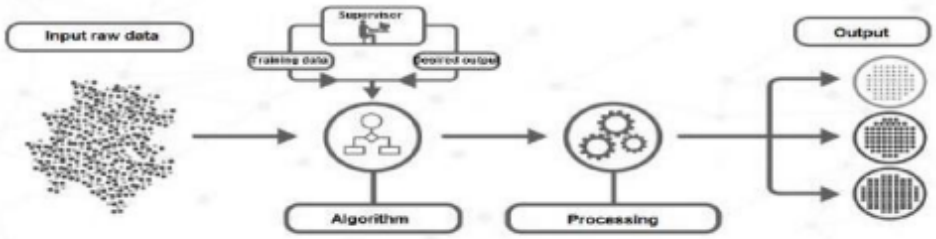


Fig 3: Supervised Learning

Note: 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.

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".

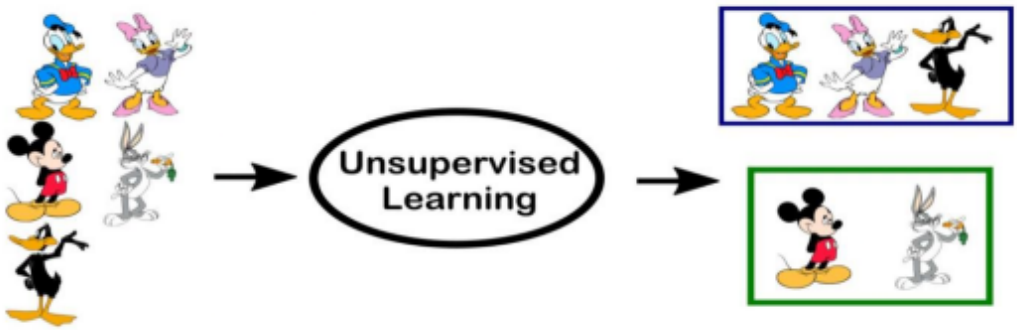
gender	age	label
M	48	sick
M	67	sick
F	53	healthy
M	49	healthy
F	34	sick
M	21	healthy

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

Unsupervised learning

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 of Unsupervised Learning

Example

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

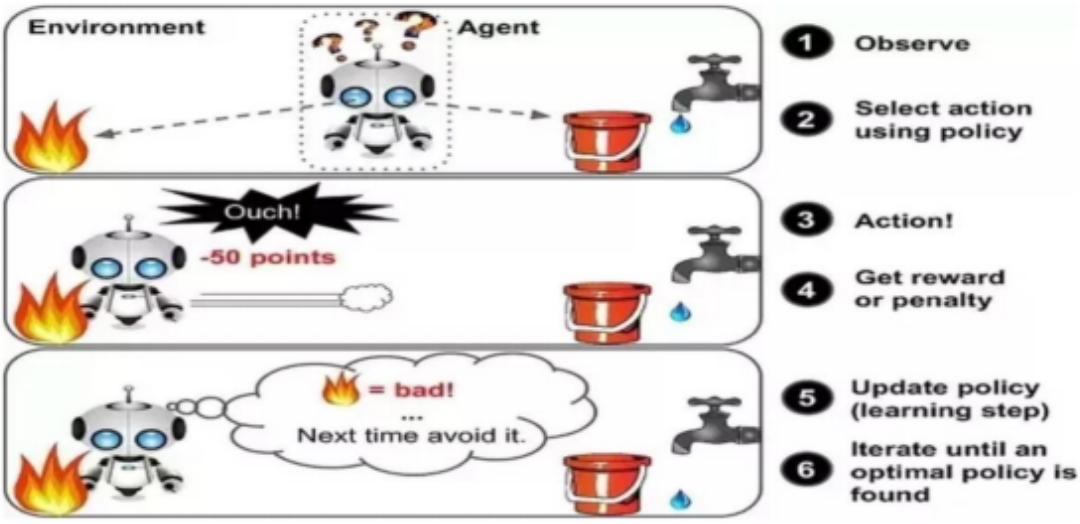
Gender	age
M	48
M	67
F	53
M	49
F	34
M	21

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

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.



Example of Reinforcement Learning

Noise:

Real-world data, which is used to feed data mining algorithms, has a number of factors that can influence it. The existence of noise is a major factor in both of these problems. It's an inevitable problem, but one that a data-driven organization must fix. Humans are prone to making mistakes when collecting data, and data collection instruments may be unreliable, resulting in dataset errors.

The errors are referred to as noise. Data noise in machine learning can cause problems since the algorithm interprets the noise as a pattern and can start generalizing from it. A noisy dataset will wreak havoc on the entire analysis pipeline. Noise can be measured as a signal to noise ratio by analysts and data scientists. As a result, by using an algorithm, any data scientist must deal with the noise in data science.

Types of Error:

The difference between the model's predicted response value and the actual observed response value from the in-sample data is called the residual for each point, and residuals refers collectively to all of the differences between all predicted and actual values.

Each out-of-sample (new/test data) difference is called a prediction error instead of residual.

In all four cases, true or false refers to whether the actual class matched the predicted class, and positive or negative refers to which classification was assigned to an observation by the model.

True positives, false positives (Type 1 error):

- **A true positive** is an outcome where the model ***correctly*** **predicts** the *positive* class.
- A **false positive** is an outcome where the model ***incorrectly*** **predicts** the *positive* class.
- Example:
  - If a model predicts an incoming email as being spam, and it really is spam, then that's considered a true positive.
  - Positive since the model predicted spam (the positive class), and true because the actual class matched the prediction.
  - Conversely, if an incoming email is labeled spam when it's actually not spam, it is considered a false positive.
- If we're building a fingerprint verification system, our model has to be more strict against a false accept (false positive) than against a false reject (false positive) because letting a criminal in has a more expensive cost than denying entrance to an employee.

True negatives, false negatives (Type 2 error):

- A true negative is an outcome where the model correctly predicts the negative class. i.e correctly rejected.
- A false negative is an outcome where the model incorrectly predicts the negative class. i.e incorrectly rejected
- Medical testing example :
  - True positive : sick people correctly diagnosed as sick
  - False Positive : Healthy people incorrectly identified as sick
  - True negative : Healthy people correctly identified as healthy.
  - False negative : Sick people incorrectly identified as healthy.

- If we're dealing with a cancer prediction model, a false negative has a higher cost than a false positive, because predicting that a patient isn't sick when in reality he has cancer can be fatal to the patient for not detecting the disease earlier.

Error Measure:

- Error measures are a tool in ML that quantify the question "how wrong was our estimation".
- It is a function that compares the output of a learned hypothesis with the output of the real target function.
- What this means in practice is that we compare the prediction of our model with the real value in data.
- An error measure is expressed as E(h, f) (a hypothesis h ∈ H, and fis the target function).
- E is almost always pointwise. It is defined by the difference at two points, therefore, we use the pointwise definition of the error measure e() to compute this error in the different points: e(h(x), f(x)).
- examples:

  squared error: $e(h(x), f(x)) = (h(x)- f(x))^2$

  binary error: $e(h(x), f(x)) = [\![h(x) \neq f(x)]\!]$ (the number of wrong classifications)
- This means that the overall error over some data E(h, f) is the average of pointwise errors e(h(x), f(x)).
- Common error measures are,
  1. Mean squared error (or root mean squared error) – Continuous data, sensitive to outliers

     Mean squared error (MSE):

     $$\frac{1}{n}\sum_{i=1}^{n}(Prediction_i - Truth_i)^2$$

     Root mean squared error (RMSE):

     $$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Prediction_i - Truth_i)^2}$$

  2. Median absolute deviation – Continuous data, often more robust
  3. Sensitivity (recall) – if you want few missed positives
  4. Specificity – if you want few negatives called positives
  5. Accuracy – Weights false positive / negatives equally
  6. Concordance – one example is kappa

Generalization

How well a model trained on the training set predicts the right output for new instances is called generalization. Generalization refers to how well the concepts learned by a machine learning model apply to specific examples not seen by the model when it was learning. The goal of a good machine learning model is to generalize well from the training data to any data from the problem domain. This allows us to make predictions in the future on data the model has never seen. Overfitting and underfitting are the two biggest causes for poor performance of machine learning algorithms. The model should be selected having the best generalization. This is said to be the case if these problems are avoided.

- **Underfitting**: Underfitting is the production of a machine learning model that is not complex enough to accurately capture relationships between a datasetâA˘ Zs features and a target variable.
- **Overfitting**; Overfitting is the production of an analysis which corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably.

Example 1:

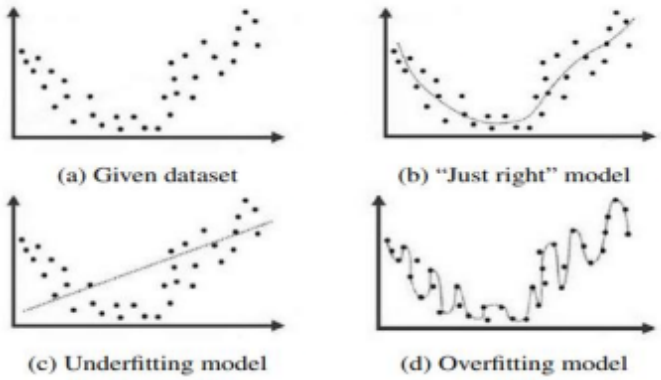


Fig 4: Examples for overfitting and overfitting models

Consider a dataset shown in Figure 4(a). Let it be required to fit a regression model to the data. The graph of a model which looks "just right" is shown in Figure 4(b). In Figure 4(c) we have a linear regression model for the same dataset and this model does seem to capture the essential features of the dataset. So this model suffers from underfitting. In Figure 4(d) we have a regression model which corresponds too closely to the given dataset and hence it does not account for small random noises in the dataset. Hence it suffers from overfitting.

Need of Machine Learning

- Machine learning has become increasingly important in recent years due to several reasons, including:

1. **Handling Big Data**: With the explosion of data in various fields, from social media to scientific research, traditional methods of data analysis have become inadequate. Machine learning algorithms can quickly and efficiently process large datasets, identify patterns, and make predictions, allowing for more accurate and effective decision-making.
2. **Automation**: Machine learning has enabled automation of many tasks that were previously performed manually, such as image and speech recognition, language translation, and even self-driving cars. This has led to increased efficiency and productivity, while also freeing up human workers to focus on more complex and creative tasks.
3. **Personalization**: Machine learning has enabled personalized experiences in various fields, such as online shopping, entertainment, and healthcare. By analyzing user data and behavior, machine learning algorithms can provide personalized recommendations, suggestions, and treatment plans, improving the overall user experience.
4. **Improved Accuracy**: Machine learning algorithms can often achieve higher accuracy than traditional statistical models in tasks such as prediction, classification, and clustering. This is because they can

learn from large amounts of data and adapt to changing circumstances, resulting in more accurate and reliable predictions.

5. **Cost-Effective**: Machine learning has become more accessible and cost-effective, with the development of open-source libraries and cloud-based services. This has allowed smaller organizations and startups to leverage the benefits of machine learning without significant upfront costs.

Data and tools

- Data and tools are essential components of machine learning.
- **Data**:
  - Data is the raw material that machine learning algorithms use to learn patterns, make predictions, and perform other tasks.
  - Good quality data is essential for machine learning, as the accuracy and effectiveness of the algorithms depend on the quality and quantity of the data used to train them.
  - In machine learning, data is typically divided into training, validation, and testing sets.
  - The training set is used to train the machine learning algorithm, while the validation set is used to evaluate the performance of the algorithm during training. The testing set is used to evaluate the performance of the final model.
- **Tools**:
  - Tools are software programs and libraries that are used to implement machine learning algorithms and analyze data. There are several popular tools for machine learning, including:

1. **Python**: Python is a popular programming language for machine learning, with many libraries and frameworks such as NumPy, Pandas, and Scikit-Learn, which provide data analysis, machine learning algorithms, and data visualization capabilities.
2. **R**: R is a programming language and environment for statistical computing and graphics, widely used in machine learning and data science. It provides many packages for data manipulation, visualization, and machine learning.
3. **TensorFlow:** TensorFlow is an open-source machine learning library developed by Google, which provides a wide range of machine learning algorithms and tools for deep learning, natural language processing, and computer vision.
4. **PyTorch:** PyTorch is an open-source machine learning library developed by Facebook, which provides a flexible platform for building and training machine learning models, including deep neural networks.
5. **Jupyter Notebook:** Jupyter Notebook is an open-source web application that allows users to create and share documents that contain live code, equations, visualizations, and narrative text. It is widely used in data science and machine learning for data exploration, experimentation, and visualization.

Review of statistics

- Statistics is the study of the collection, analysis, interpretation, presentation, and organization of data. It is a crucial tool in various fields, including science, social sciences, business, and engineering, among others.

- Here are some of the key aspects of statistics:
  1. **Descriptive Statistics**: Descriptive statistics is the process of summarizing and describing the properties of data, such as measures of central tendency (mean, median, mode) and measures of variability (standard deviation, variance).
  2. **Inferential Statistics**: Inferential statistics involves making predictions and generalizations about a population based on a sample of data. It uses probability theory and hypothesis testing to determine whether an observed effect is statistically significant.
  3. **Probability Theory**: Probability theory is the branch of mathematics that deals with the study of random events and their likelihood of occurrence. It is used in statistics to model and predict the outcomes of experiments and observations.
  4. **Sampling:** Sampling involves selecting a subset of data from a larger population for analysis. In statistics, the quality of the analysis depends on the representativeness of the sample and the sampling method used.
  5. **Regression Analysis**: Regression analysis is a statistical method used to establish a relationship between a dependent variable and one or more independent variables. It is commonly used in machine learning for predictive modeling.
  6. **Hypothesis Testing**: Hypothesis testing is a statistical method used to determine whether a hypothesis about a population is true or false. It is used in machine learning to evaluate the significance of model performance and determine whether the model is valid.
  7. **Statistical Learning**: Statistical learning is a type of machine learning that uses statistical methods to analyze data and make predictions. It includes techniques such as linear regression, logistic regression, decision trees, and random forests.
- In the machine learning perspective, statistics is a critical component of data analysis, model selection, and evaluation. Machine learning algorithms rely on statistical techniques to learn from data and make accurate predictions. Understanding statistics is therefore essential for anyone working in machine learning.

Training, Validation and Test data

- In machine learning, the process of training a model involves using a dataset to teach the model to recognize patterns and make predictions.
- However, it is important to evaluate the performance of the model on new, unseen data to ensure that it generalizes well and is not overfitting to the training data.
- To achieve this, the dataset is often split into three subsets: training, validation, and test data.
  1. **Training Data**: The training data is used to train the model, and it typically represents the largest portion of the dataset. The model learns to recognize patterns in the training data and adjust its parameters accordingly.
  2. **Validation Data**: The validation data is used to evaluate the performance of the model during the training process. It is used to fine-tune the model's hyper parameters, such as the learning rate or regularization strength, to improve its performance on the validation set.

3. **Test Data**: The test data is used to evaluate the final performance of the model after the training process is complete. It represents new, unseen data, and it is used to estimate the model's performance on new data that it has not seen during training.

- The process of splitting the dataset into training, validation, and test data is called data partitioning.
- The size of each subset depends on the size of the overall dataset and the complexity of the model.
- In general, the training set should be larger than the validation and test sets, and the test set should be large enough to provide a reliable estimate of the model's performance. It is important to ensure that the three subsets are representative of the overall dataset to avoid bias in the model's performance.

Theory of learning

- The theory of learning in machine learning is concerned with understanding how machines can learn from data to improve their performance on a given task.
- Here are some key concepts in the theory of learning:
  1. Feasibility of Learning: Feasibility of learning refers to the question of whether a particular learning problem can be solved algorithmically. Some learning problems are inherently difficult and cannot be solved efficiently, while others can be solved with relatively simple algorithms.
  2. Error and Noise: In machine learning, error refers to the difference between the predicted output of a model and the true output. Noise refers to the random variation in the data that is not related to the underlying patterns that the model is trying to learn. To build a good model, it is important to reduce both error and noise in the data.
  3. Training versus Testing: Training refers to the process of using a dataset to teach a machine learning model to recognize patterns and make predictions. Testing refers to the process of evaluating the model's performance on new, unseen data. It is important to test the model on new data to ensure that it generalizes well and is not over fitting to the training data.
- In machine learning, the goal is to learn a model that can make accurate predictions on new data.
- The process of learning involves finding a function that maps inputs to outputs by minimizing the error between the predicted output and the true output.
- However, it is important to balance the complexity of the model with its ability to generalize to new data. Overly complex models can be overfit to the training data and perform poorly on new data. Therefore, it is important to evaluate the model's performance on new data to ensure that it generalizes well.

Generalization Bound:

A generalization bound – or, more precisely, a generalization error bound – is a statement about the predictive performance of a learning algorithm or class of algorithms. Here, a learning algorithm is viewed as a procedure that takes some finite training sample of labeled instances as input and returns a hypothesis regarding the labels of all instances, including those which may not have appeared in the training sample. Assuming labeled instances are drawn from some fixed distribution, the quality of a hypothesis can be measured in terms of its risk

– its incompatibility with the distribution.

The performance of a learning algorithm can then be expressed in terms of the expected risk of its hypotheses given randomly generated training samples.

- The generalization bound is a theoretical guarantee that provides an upper bound on the expected difference between the true error and the empirical error of a learning algorithm on new, unseen data.
- The generalization bound is an important theoretical concept in machine learning, as it provides a way to bound the expected error of a learning algorithm on new, unseen data.
    - Suppose you are training a binary classifier to predict whether an email is spam or not. You have a dataset with 100 emails, of which 60 are spam and 40 are not. You split the dataset into a training set with 80 emails and a test set with 20 emails. After training the model on the training set, you get an accuracy of 90%.
    - The generalization bound in this case provides an upper bound on the difference between the true error (i.e., the error on new, unseen data) and the empirical error (i.e., the error on the training data) of the model.
    - For example, the generalization bound might show that the expected error on new data is no more than 5% higher than the error on the training data, which means that the expected accuracy on the test set would be at least 85%.

Approximation-generalization tradeoff

- The approximation-generalization tradeoff is a fundamental challenge in machine learning, where the goal is to find a model that is both complex enough to capture the underlying patterns in the data and simple enough to generalize well to new, unseen data. Simpler models may generalize better, but they may not be able to capture the complexity of the underlying patterns in the data.
- The approximation-generalization tradeoff is a key challenge in machine learning, as finding the right balance between model complexity and generalization is crucial for building accurate and robust models.
- Suppose you are training a linear regression model to predict the price of a house based on its size in square feet. A simple linear model might be too simplistic to capture the complex patterns in the data, while a more complex model such as a polynomial regression might overfit to the training data and generalize poorly.
- The tradeoff here is between model complexity and generalization performance. For example, you might try different degrees of polynomial regression (i.e., different levels of complexity) and plot the training error and the test error as a function of the degree of the polynomial. The plot would help you find the right balance between underfitting and overfitting.

Bias and variance

- Bias and variance are two sources of error in machine learning. Bias refers to the error that arises from approximating a real-world problem with a simpler model. Variance refers to the error that arises from the model's sensitivity to small fluctuations in the training data. A high-bias model is too simple and may underfit the data, while a high-variance model is too complex and may overfit the data.

- Bias and variance are two important sources of error in machine learning, and understanding how to control them is critical for building models that generalize well.
- Consider a supervised learning problem where the goal is to predict the age of a person based on their height and weight. If we use a linear model, we might introduce bias into the model by assuming a linear relationship between height, weight, and age, when in reality the relationship might be more complex. On the other hand, if we use a very complex model such as a deep neural network, we might introduce variance into the model by fitting the noise in the training data and overfitting to the training data.
- To find the right balance between bias and variance, you might try different models with different levels of complexity and plot the training error and the test error as a function of the complexity.

Learning curve

- A learning curve is a plot of the model's performance on the training data and the validation data as a function of the number of training examples. Learning curves can help diagnose problems with the model, such as underfitting or overfitting, and can help determine the optimal size of the training dataset and the optimal complexity of the model.
- Learning curves can provide valuable insights into the performance of a model, and can help determine the optimal number of training examples and the optimal complexity of the model.
- Consider a classification problem where the goal is to predict whether a tumor is malignant or benign based on various features such as size, shape, and texture. A learning curve might plot the performance of a model on the training data and the validation data as a function of the number of training examples.
- For example, you might start with a small dataset of 10 tumors and plot the training error and the validation error as a function of the number of examples. The plot might show that the model is under fitting the data, which means that you need to increase the complexity of the model or collect more training data. Then, you might increase the size of the dataset to 50 tumors and plot the learning curve again. The plot might show that the model is now over fitting the data, which means that you need to decrease the complexity of the model or use regularization techniques.

Testing generalization:

Cross-validation

We can measure the generalization ability of a hypothesis, namely, the quality of its inductive bias, if we have access to data outside the training set. We simulate this by dividing the training set we have into two parts. We use one part for training (that is, to find a hypothesis), and the remaining part is called the validation set and is used to test the generalization ability. Assuming large enough training and validation sets, the hypothesis that is the most accurate on the validation set is the best one (the one that has the best inductive bias). This process is called cross-validation.

UNIT 2

SUPERVISED LEARNING

***Classification Problems*:**

In machine learning, classification is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.

Example

Consider the following data:

Table 1.1: Example data for a classification problem

Score1	29	22	10	31	17	33	32	20
Score2	43	29	47	55	18	54	40	41
Result	Pass	Fail	Fail	Pass	Fail	Pass	Pass	Pass

Data in Table 1.1 is the training set of data. There are two attributes "Score1" and "Score2". The class label is called "Result". The class label has two possible values "Pass" and "Fail". The data can be divided into two categories or classes: The set of data for which the class label is "Pass" and the set of data for which the class label is"Fail". Let us assume that we have no knowledge about the data other than what is given in the table. Now, the problem can be posed as follows: If we have some new data, say "Score1 = 25" and "Score2 = 36", what value should be assigned to "Result" corresponding to the new data; in other words, to which of the two categories or classes the new observation should be assigned?

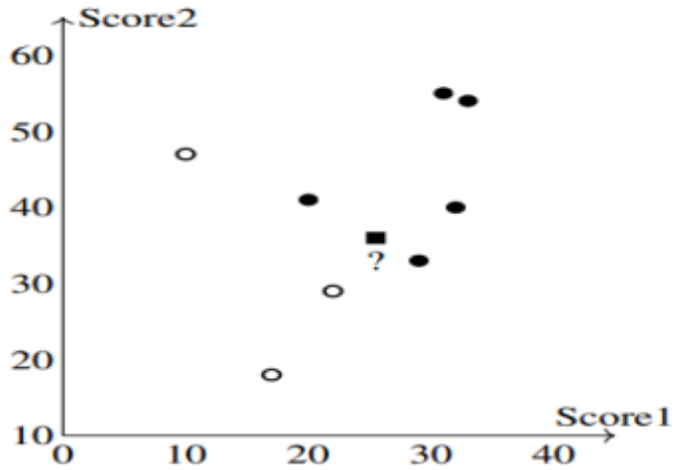


Fig 5: Graphical representation of data in Table 1.1. Solid dots represent data in "Pass" class and hollow dots data in "Fail" class. The class label of the square dot is to be determined.

To answer this question, using the given data alone we need to find the rule, or the formula, or the method that has been used in assigning the values to the class label "Result". The problem of finding this rule or formula or the method is the classification problem. In general, even the general form of the rule or function or method will not be known. So several different rules, etc. may have to be tested to obtain the correct rule or function or method.

Regression:

A regression problem is the problem of determining a relation between one or more independent variables and an output variable which is a real continuous variable, given a set of observed values of the set of independent variables and the corresponding values of the output variable.

Examples

1. Let us say we want to have a system that can predict the price of a used car. Inputs are the car attributes âA˜T brand, year, engine capacity, mileage, and other information â ˘ A˜T that we ˘ believe affect a car's worth. The output is the price of the car.
2. Consider the navigation of a mobile robot, say an autonomous car. The output is the angle by which the steering wheel should be turned at each time, to advance without hitting obstacles and deviating from the route. Inputs are provided by sensors on the car like a video camera, GPS, and so forth.
3. In finance, the capital asset pricing model uses regression for analyzing and quantifying the systematic risk of an investment.
4. In economics, regression is the predominant empirical tool. For example, it is used to predict consumption spending, inventory investment, purchases of a country's exports, spending on imports, labor demand, and labor supply.

Simple linear regression

Let x be the independent predictor variable and y the dependent variable. Assume that we have a set of observed values of x and y:

A simple linear regression model defines the relationship between x and y using a line defined by an equation in the following form:

$$y = \alpha + \beta x$$

In order to determine the optimal estimates of α and β, an estimation method known as Ordinary Least Squares (OLS) is used. In the OLS method, the values of y-intercept and slope are chosen such that they minimize the sum of the squared errors; that is, the sum of the squares of the vertical distance between the predicted y-value and the actual y-value (see Figure 6). Let yˆi be the predicted value of yi. Then the sum of squares of errors is given by

$$E = \sum_{i=1}^{n} (yi - \hat{y}i)^2$$

$$E = \sum_{i=1}^{n} ([yi - (\alpha + \beta xi)]2$$

So we are required to find the values of α and β such that E is minimum. Using methods of calculus, we can show that the values of a and b, which are respectively the values of α and β for which E is minimum, can be obtained by solving the equations.(**refer to class notes**)

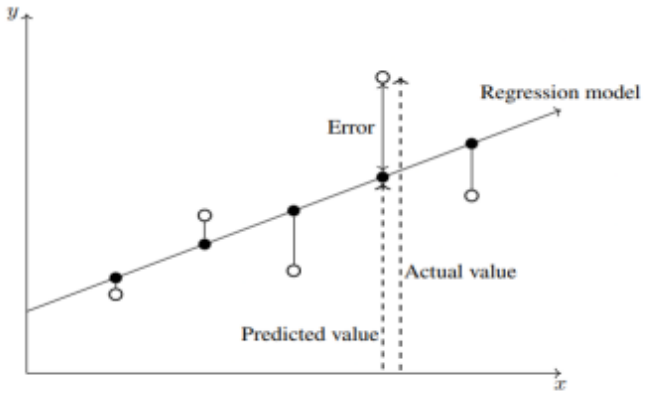


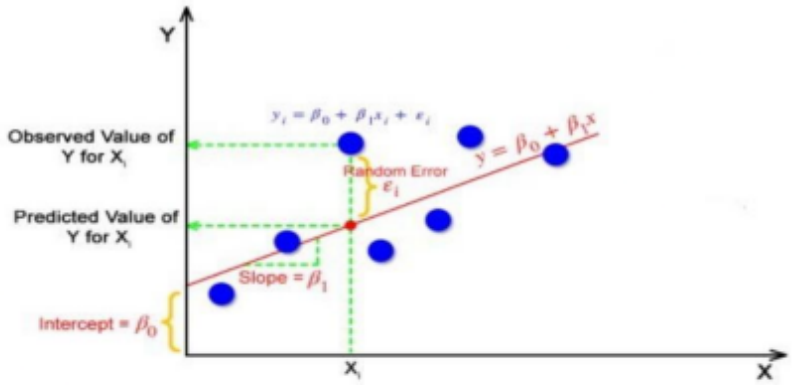
Fig 6: Errors in observed values

Finding best fit line with linear regression:

To calculate best-fit line linear regression uses a traditional slope-intercept form which is given below,

$$Yi = \beta 0 + \beta 1 Xi$$

where Yi = Dependent variable, β0 = constant/Intercept, β1 = Slope/Intercept, Xi = Independent variable.



This algorithm explains the linear relationship between the dependent (output) variable y and the independent (predictor) variable X using a straight line Y= B0 + B1 X.

But how the linear regression finds out which is the best fit line?

The goal of the linear regression algorithm is to get the best values for B0 and B1 to find the best fit line. The best fit line is a line that has the least error which means the error between predicted values and actual values should be minimum.

Random Error (Residuals)

In regression, the difference between the observed value of the dependent variable(yi) and the predicted value(predicted) is called the residuals.

$$\varepsilon i = y \text{ predicted} - yi$$

where y predicted = B0 + B1 Xi

What is the best fit line?

In simple terms, the best fit line is a line that fits the given scatter plot in the best way. Mathematically, the best fit line is obtained by minimizing the Residual Sum of Squares (RSS).

Cost Function for Linear Regression

The cost function helps to work out the optimal values for B0 and B1, which provides the best fit line for the data points. In Linear Regression, generally Mean Squared Error (MSE) cost function is used, which is the average of squared error that occurred between the ypredicted and yi.

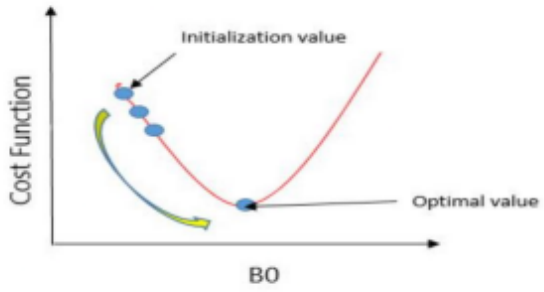
We calculate MSE using simple linear equation y=mx+b:

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - (B1x_i + B0))^2$$

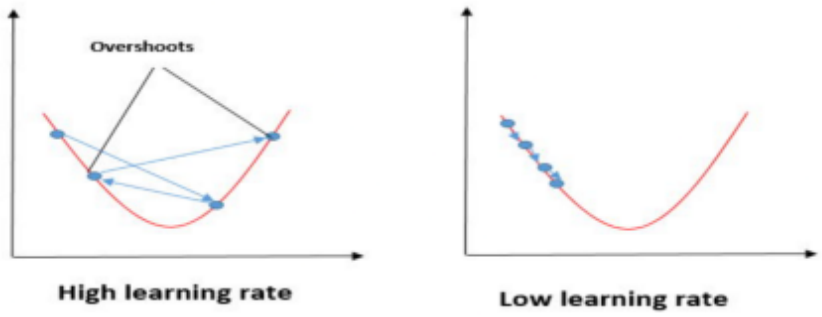
Using the MSE function, we'll update the values of B0 and B1 such that the MSE value settles at the minima. These parameters can be determined using the gradient descent method such that the value for the cost function is minimum.

Gradient Descent for Linear Regression

Gradient Descent is one of the optimization algorithms that optimize the cost function(objective function) to reach the optimal minimal solution. To find the optimum solution we need to reduce the cost function(MSE) for all data points. This is done by updating the values of B0 and B1 iteratively until we get an optimal solution. A regression model optimizes the gradient descent algorithm to update the coefficients of the line by reducing the cost function by randomly selecting coefficient values and then iteratively updating the values to reach the minimum cost function.



Let's take an example to understand this. Imagine a U-shaped pit. And you are standing at the uppermost point in the pit, and your motive is to reach the bottom of the pit. Suppose there is a treasure at the bottom of the pit, and you can only take a discrete number of steps to reach the bottom. If you opted to take one step at a time, you would get to the bottom of the pit in the end but, this would take a longer time. If you decide to take larger steps each time, you may achieve the bottom sooner but, there's a probability that you could overshoot the bottom of the pit and not even near the bottom. In the gradient descent algorithm, the number of steps you're taking can be considered as the learning rate, and this decides how fast the algorithm converges to the minima.



To update B0 and B1, we take gradients from the cost function. To find these gradients, we take partial derivatives for B0 and B1.

$$J = \frac{1}{n}\sum_{i=1}^{n}(B_0 + B_1 \cdot x_i - y_i)^2$$

$$\frac{\partial J}{\partial B_0} = \frac{2}{n}\sum_{i=1}^{n}(B_0 + B_1 \cdot x_i - y_i)$$

$$\frac{\partial J}{\partial B_1} = \frac{2}{n}\sum_{i=1}^{n}(B_0 + B_1 \cdot x_i - y_i) \cdot x_i$$

$$J = \frac{1}{n}\sum_{i=1}^{n}(B_0 + B_1 \cdot x_i - y_i)^2$$

$$\frac{\partial J}{\partial B_0} = \frac{2}{n}\sum_{i=1}^{n}(B_0 + B_1 \cdot x_i - y_i)$$

$$\frac{\partial J}{\partial B_1} = \frac{2}{n}\sum_{i=1}^{n}(B_0 + B_1 \cdot x_i - y_i) \cdot x_i$$

$$B_0 = B_0 - \alpha \cdot \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i)$$

$$B_1 = B_1 - \alpha \cdot \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i) \cdot x_i$$

We need to minimize the cost function J. One of the ways to achieve this is to apply the batch gradient descent algorithm. In batch gradient descent, the values are updated in each iteration. (Last two equations shows the updating of values) The partial derivates are the gradients, and they are used to update the values of B0 and B1. Alpha is the learning rate.

Evaluation Metrics for Linear Regression

The strength of any linear regression model can be assessed using various evaluation metrics. These evaluation metrics usually provide a measure of how well the observed outputs are being generated by the model.

The most used metrics are,

Coefficient of Determination or R-Squared (R2)

Root Mean Squared Error (RSME) and Residual Standard Error (RSE)

Coefficient of Determination or R-Squared (R2)

R-Squared is a number that explains the amount of variation that is explained/captured by the developed model. It always ranges between 0 & 1 . Overall, the higher the value of R-squared, the better the model fits the data.

Mathematically it can be represented as,

$$R2 = 1 - (RSS/TSS)$$

Residual sum of Squares (RSS) is defined as the sum of squares of the residual for each data point in the plot/data. It is the measure of the difference between the expected and the actual observed output.

$$RSS = \sum_{i=1}^{n}(y_i - b_0 - b_1 x_i)^2$$

Total Sum of Squares (TSS) is defined as the sum of errors of the data points from the mean of the response variable. Mathematically TSS is, Total Sum of Squares

$$TSS = \sum (y_i - \bar{y}_i)^2$$

where y hat is the mean of the sample data points.

Root Mean Squared Error

The Root Mean Squared Error is the square root of the variance of the residuals. It specifies the absolute fit of the model to the data i.e. how close the observed data points are to the predicted values. Mathematically it can be represented as,

$$RMSE = \sqrt{\frac{RSS}{n}} = \sqrt{\sum_{i=1}^{n} (y_i^{Actual} - y_i^{Predicted})^2 / n}$$

To make this estimate unbiased, one has to divide the sum of the squared residuals by the degrees of freedom rather than the total number of data points in the model. This term is then called the Residual Standard Error (RSE). Mathematically it can be represented as,

$$RSE = \sqrt{\frac{RSS}{df}} = \sqrt{\sum_{i=1}^{n} (y_i^{Actual} - y_i^{Predicted})^2 / (n-2)}$$

Linear Regression Vs Logistic Regression

Linear Regression	Logistic Regression
Used to solve regression problems	Used to solve classification problems
The response variables are continuous in nature	The response variable is categorical in nature
It helps estimate the dependent variable when there is a change in the independent variable	It helps to calculate the possibility of a particular event taking place
It is a straight line	It is an S-curve (S = Sigmoid)

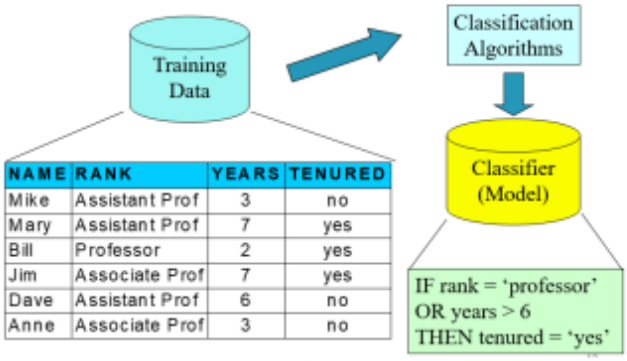
Classification vs. Prediction

- Classification
  - predicts categorical class labels
  - classifies data based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Prediction
  - predicts unknown or missing values
- Typical applications
  - Credit/loan approval
  - Target marketing
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

Classification—A Two-Step Process

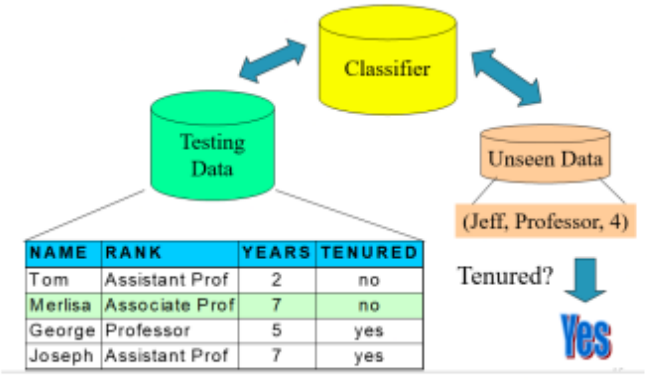
- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

Process (1): Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Process (2): Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Issues: Data Preparation

- Data cleaning
  - Preprocess data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes
- Data transformation
  - Generalize and/or normalize data

Issues: Evaluating Classification Methods

- Accuracy
  - classifier accuracy: predicting class label
  - predictor accuracy: guessing value of predicted attributes
- Speed
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

Decision Tree Representation:

Decision tree learning is a method for approximating discrete valued target functions, in which the learned function is represented by a decision tree. Decision tree learning is one of the most widely used and practical methods for inductive inference.

Example

Consider the following situation. Somebody is hunting for a job. At the very beginning, he decides that he will consider only those jobs for which the monthly salary is at least Rs.50,000. Our job hunter does not like spending much time traveling to place of work. He is comfortable only if the commuting time is less than one hour. Also, he expects the company to arrange for a free coffee every morning! The decisions to be made before deciding to accept or reject a job offer can be schematically represented as in Figure 7

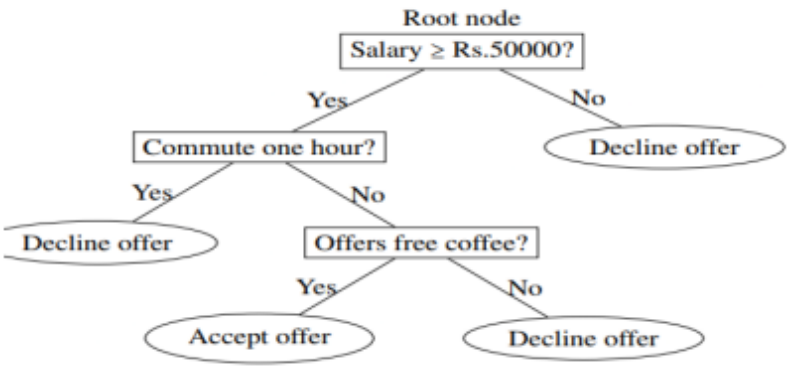


Fig 7: Example of Decision Tree

Appropriate Problem for Decision Tree Learning:

Decision tree learning is a versatile machine learning technique that can be applied to a wide range of problems, including classification and regression tasks. Decision trees are particularly useful for problems where we want to understand and visualize the decision-making process. Some appropriate problem domains for decision tree learning:

1. Classification Problems
2. Regression Problems

3. Anomaly Detection
4. Customer Segmentation
5. Feature Selection
6. Multi-class Problems
7. Interpretable Models
8. Recommendation Systems
9. Resource Allocation
10. Education

Basic Decision tree algorithm:

Outline

1. Place the "best" feature (or, attribute) of the dataset at the root of the tree.
2. Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for a feature.
3. Repeat Step 1 and Step 2 on each subset until we find leaf nodes in all the branches of the tree.

ID3 (Iterative Dichotomiser 3):

Assumptions

- The algorithm uses information gain to select the most useful attribute for classification.
- We assume that there are only two class labels, namely, "+" and "–". The examples with class labels "+" are called positive examples and others negative examples.

Notations

The following notations are used in the algorithm:

S The set of examples

C The set of class labels

F The set of features

A An arbitrary feature (attribute)

Values(A) The set of values of the feature A

v An arbitrary value of A

Sv The set of examples with A = v

Root The root node of a tree

Algorithm:

ID3(S, F, C)

1. Create a root node for the tree.
2. if (all examples in S are positive) then
3. return single node tree Root with label "+"
4. end if
5. if (all examples are negative) then

6. return single node tree Root with label "–"
7. end if
8. if (number of features is 0) then
9. return single node tree Root with label equal to the most common class label.
10. Else
11. Let A be the feature in F with the highest information gain.
12. Assign A to the Root node in decision tree.
13. for all (values v of A) do
14. Add a new tree branch below Root corresponding to v.
15. if (Sv is empty) then
16. Below this branch add a leaf node with label equal to the most common class label in the set S.
17. Else
18. Below this branch add the subtree formed by applying the same algorithm ID3 with the values ID3(Sv, C, F − {A}).
19. end if
20. end for
21. end if

Entropy

The degree to which a subset of examples contains only a single class is known as purity, and any subset composed of only a single class is called a pure class. Informally, entropy3 is a measure of "impurity" in a dataset. Sets with high entropy are very diverse and provide little information about other items that may also belong in the set, as there is no apparent commonality. Entropy is measured in bits. If there are only two possible classes, entropy values can range from 0 to 1. For n classes, entropy ranges from 0 to log2 (n). In each case, the minimum value indicates that the sample is completely homogeneous, while the maximum value indicates that the data are as diverse as possible, and no group has even a small plurality.

Consider a segment S of a dataset having c number of class labels. Let pi be the proportion of examples in S having the i th class label. The entropy of S is defined as

Entropy (S) = c $\sum$ i=1 −pi log2 (pi).

Information gain

Let S be a set of examples, A be a feature (or, an attribute), Sv be the subset of S with A = v, and Values (A) be the set of all possible values of A. Then the information gain of an attribute A relative to the set S, denoted by Gain (S, A), is defined as

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|Sv|}{|S|} \times \text{Entropy}(Sv).$$

where |S| denotes the number of elements in S

Example 1: Use ID3 algorithm to construct a decision tree for the data in Table

Day	outlook	temperature	humidity	wind	playtennis
D1	sunny	hot	high	weak	no
D2	sunny	hot	high	strong	no
D3	overcast	hot	high	weak	yes
D4	rain	mild	high	weak	yes
D5	rain	cool	normal	weak	yes
D6	rain	cool	normal	strong	no
D7	overcast	cool	normal	strong	yes
D8	sunny	mild	high	weak	no
D9	sunny	cool	normal	weak	yes
D10	rain	mild	normal	weak	yes
D11	sunny	mild	normal	strong	yes
D12	overcast	mild	high	strong	yes
D13	overcast	hot	normal	weak	yes
D14	rain	mild	high	strong	no

Solution:

Note that, in the given data, there are four features but only two class labels (or, target variables), namely, "yes" and "no".

Step 1

We first create a root node for the tree

Step 2

Note that not all examples are positive (class label "yes") and not all examples are negative (class label "no"). Also the number of features is not zero.

Step 3

We have to decide which feature is to be placed at the root node. For this, we have to calculate the information gains corresponding to each of the four features. The computations are shown below.

(i) Calculation of Entropy (S)

$$\text{Entropy (S)} = -p_{yes}\log_2(p_{yes}) - p_{no}\log_2(p_{no})$$
$$= -(9/14) \times \log_2(9/14) - (5/14) \times \log_2(5/14)$$
$$= 0.9405$$

(ii) Calculation of Gain (S, outlook)

The values of the attribute "outlook" are "sunny", " overcast" and "rain". We have to calculate Entropy (Sv) for v = sunny, v = overcast and v = rain

$$\text{Entropy (Ssunny)} = -(3/5) \times \log_2(3/5) - (2/5) \times \log_2(2/5) = 0.9710$$

$$\text{Entropy (Sovercast)} = -(4/4) \times \log_2(4/4) = 0$$

$$\text{Entropy (Srain)} = -(3/5) \times \log_2(3/5) - (2/5) \times \log_2(2/5) = 0.9710$$

$$\text{Gain (S, outlook)} = \text{Entropy (S)} - \frac{|S_{sunny}|}{|S|} \times \text{Entropy (Ssunny)} - \frac{|S_{overcast}|}{|S|} \times \text{Entropy (Sovercast)} - \frac{|S_{rain}|}{|S|} \times \text{Entropy (Srain)}$$

$$= 0.9405 - (5/14) \times 0.9710 - (4/14) \times 0 - (5/14) \times 0.9710 = 0.2469$$

(iii) Calculation of Gain (S, temperature)

The values of the attribute "temperature" are "hot", "mild" and "cool". We have to calculate Entropy (Sv) for v = hot, v = mild and v = cool

$$\begin{aligned}
\text{Entropy}\,(S_{\text{hot}}) &= -(2/4) \times \log_2(2/4) - (2/4) \times \log_2(2/4) \\
&= 1.0000 \\
\text{Entropy}\,(S_{\text{mild}}) &= -(4/6) \times \log_2(4/6) - (2/6) \times \log_2(2/6) \\
&= 0.9186 \\
\text{Entropy}\,(S_{\text{cool}}) &= -(3/4) \times \log_2(3/4) - (1/4) \times \log_2(1/4) \\
&= 0.8113 \\
\text{Gain}\,(S,\ \text{temperature}) &= \text{Entropy}\,(S) - \frac{|S_{\text{hot}}|}{|S|} \times \text{Entropy}\,(S_{\text{hot}}) \\
&\quad - \frac{|S_{\text{mild}}|}{|S|} \times \text{Entropy}\,(S_{\text{mild}}) \\
&\quad - \frac{|S_{\text{cool}}|}{|S|} \times \text{Entropy}\,(S_{\text{cool}}) \\
&= 0.9405 - (4/14) \times 1.0000 - (6/14) \times 0.9186 \\
&\quad - (4/14) \times 0.8113 \\
&= 0.0293
\end{aligned}$$

(iv) Calculation of Gain (S, humidity) and Gain (S, wind)

The following information gains can be calculated in a similar way:

Gain (S, humidity) = 0.151

Gain (S, wind) = 0.048

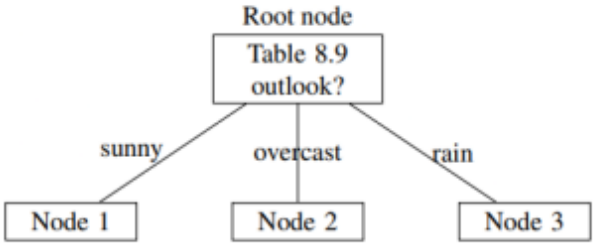
Step 4

We find the highest information gain which is the maximum among Gain(S, outlook), Gain(S, temperature), Gain(S, humidity) and Gain(S, wind). Therefore, we have:

highest information gain = max{0.2469, 0.0293, 0.151, 0.048}

= 0.2469

This corresponds to the feature "outlook". Therefore, we place "outlook" at the root node



Step 5

Let S(1) = Soutlook=sunny.

Day	outlook	temperature	humidity	wind	playtennis
D1	sunny	hot	high	weak	no
D2	sunny	hot	high	strong	no
D8	sunny	mild	high	weak	no
D9	sunny	cool	normal	weak	yes
D11	sunny	mild	normal	strong	yes

$$\begin{aligned}
\text{Gain}(S^{(1)}, \text{temp}) = \text{Entropy}(S^{(1)}) &- \frac{\left|S^{(1)}_{\text{temp}=\text{hot}}\right|}{\left|S^{(1)}\right|} \times \text{Entropy}(S^{(1)}_{\text{temp}=\text{hot}}) \\
&- \frac{\left|S^{(1)}_{\text{temp}=\text{mild}}\right|}{\left|S^{(1)}\right|} \times \text{Entropy}(S^{(1)}_{\text{temp}=\text{mild}}) \\
&- \frac{\left|S^{(1)}_{\text{temp}=\text{cool}}\right|}{\left|S^{(1)}\right|} \times \text{Entropy}(S^{(1)}_{\text{temp}=\text{cool}})
\end{aligned}$$

$$= \left[-(2/5)\log_2(2/5) - (3/5)\log_2(3/5)\right] - (2/5) \times \left[-(2/2)\log(2/2))\right] - (2/5) \times \left[-(1/2)\log(1/2) - (1/2)\log_2(1/2)\right] - (1/5) \times \left[-(1/1)\log(1/1)\right] = 0.5709$$

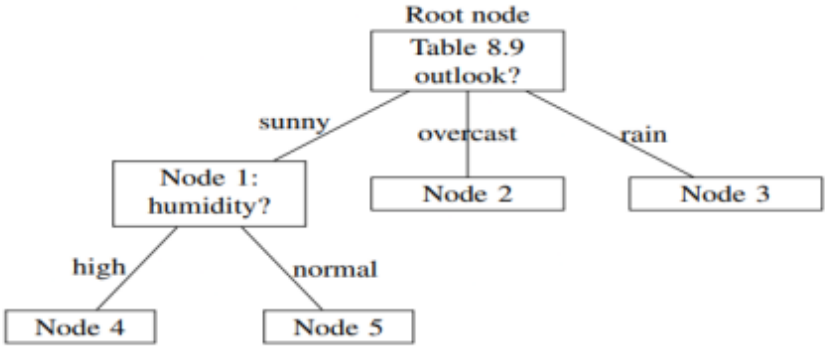
$$\text{Gain}(S^{(1)}, \text{hum}) = \text{Entropy}(S^{(1)}) - \frac{\left|S^{(1)}_{\text{hum = high}}\right|}{\left|S^{(1)}\right|} \times \text{Entropy}(S^{(1)}_{\text{hum = high}}) - \frac{\left|S^{(1)}_{\text{hum = normal}}\right|}{\left|S^{(1)}\right|} \times \text{Entropy}(S^{(1)}_{\text{hum = normal}})$$

$$= \left[-(2/5)\log_2(2/5) - (3/5)\log_2(3/5)\right] - (3/5) \times \left[-(3/3)\log(3/3))\right] - (2/5) \times \left[-(2/2)\log(2/2)\right] = 0.9709$$

$$\text{Gain}(S^{(1)}, \text{wind}) = \text{Entropy}(S^{(1)}) - \frac{\left|S^{(1)}_{\text{wind = weak}}\right|}{\left|S^{(1)}\right|} \times \text{Entropy}(S^{(1)}_{\text{wind = weak}}) - \frac{\left|S^{(1)}_{\text{wind = strong}}\right|}{\left|S^{(1)}\right|} \times \text{Entropy}(S^{(1)}_{\text{wind = strong}})$$

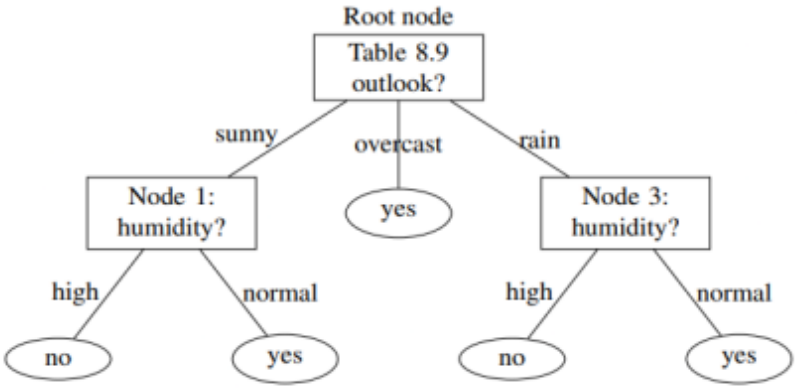
$$= \left[-(2/5)\log_2(2/5) - (3/5)\log_2(3/5)\right] - (3/5) \times \left[-(2/3)\log(2/3) - (1/3)\log_2(1/3))\right] - (2/5) \times \left[-(1/2)\log(1/2) - (1/2)\log(1/2)\right] = 0.0110$$

The maximum of Gain(S(1), temp), Gain(S(1), hum) and Gain(S(1), wind) is Gain(S(1), hum). Hence we place "humidity" at Node 1 and split this node into two branches according to the valuesof the feature "humidity" to get the tree as below:



Step 6

It can be seen that all the examples in the the data set corresponding to Node 4 have the same class label "no" and all the examples corresponding to Node 5 have the same class label "yes". So we represent Node 4 as a leaf node with value "no" and Node 5 as a leaf node with value "yes". Similarly, all the examples corresponding to Node 2 have the same class label "yes". So we convert Node 2 as a leaf node with value " yes. Finally, let S(2) = Soutlook = rain. The highest information gain for this data set is Gain(S(2), humidity). The branches resulting from splitting this node corresponding to the values "high" and "normal" of "humidity" lead to leaf nodes with class labels "no" and "yes". With these changes, we get the tree below:



Approaches to avoiding overfitting

The main approach to avoid overfitting is pruning. Pruning is a technique that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.

- We may apply pruning earlier, that is, before it reaches the point where it perfectly classifies the training data.
- We may allow the tree to overfit the data, and then post-prune the true.

Now there is the problem of what criterion is to be used to determine the correct final tree size. One commonly used criterion is to use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.

Problem of missing attributes

The missing values are indicated by "?"s. The following are some of the methods used to handle the problem of missing attributes.

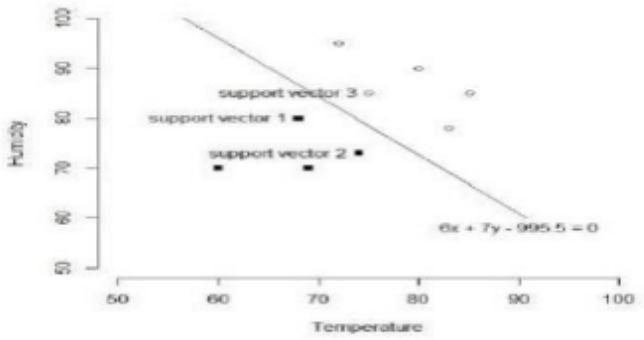
- Deleting cases with missing attribute values
- Replacing a missing attribute value by the most common value of that attribute
- Assigning all possible values to the missing attribute value
- Replacing a missing attribute value by the mean for numerical attributes
- Assigning to a missing attribute value the corresponding value taken from the closest t cases, or replacing a missing attribute value by a new value

Support Vector Machine:

SVM, or Support Vector Machine, is a supervised machine learning algorithm used for classification and regression tasks. It is a powerful and versatile algorithm that works well for both linearly and non-linearly separable data. SVMs are particularly useful when we want to find a clear boundary between different classes in your dataset.

Components of SVM:

Support Vectors: In SVM, the algorithm identifies a subset of data points as "support vectors." These are the data points that are closest to the decision boundary (the hyperplane) and have the most influence on determining the position and orientation of the hyperplane.



Hyperplane: In the context of classification, a hyperplane is a decision boundary that separates data points into different classes. For a two-class problem, the goal is to find the hyperplane that maximizes the margin (the distance between the hyperplane and the nearest data points of each class). The SVM aims to find the hyperplane that best separates the data while minimizing classification error.For a linearly separable dataset, a linear hyperplane is used.For non-linearly separable data, SVM can use kernel functions to map the data into a higher-dimensional space, where a linear hyperplane can separate the classes.

Kernel Trick: SVMs can handle non-linearly separable data by using kernel functions. These functions transform the original input space into a higher-dimensional feature space, where the data becomes linearly separable. Common kernel functions include the linear, polynomial, radial basis function (RBF), and sigmoid kernels.

Margin: The margin is the distance between the hyperplane and the nearest data point of any class. SVM aims to maximize this margin because a larger margin often leads to better generalization to unseen data and reduced overfitting.

C Parameter: SVM has a hyperparameter called 'C' that controls the trade-off between maximizing the margin and minimizing the classification error. A small 'C' allows for a larger margin but may tolerate some misclassified points, while a larger 'C' puts more emphasis on correct classification but may result in a narrower margin.

Classification: For classification tasks, SVM assigns new data points to one of the two classes based on which side of the hyperplane they fall. The sign of the distance of a point from the hyperplane determines its class (positive or negative).

Regression: SVM can also be used for regression tasks, where the goal is to predict a continuous target variable. In this case, instead of a hyperplane, SVM tries to fit a function that minimizes the error between predicted and actual values.

Linearly separable data

Consider a two-class data set having n numeric features and two possible class labels −1 and +1. Let the vector $\vec{x} = (x1, \ldots, xn)$ represent the values of the features in one instance of the data set. We say that the data set is linearly separable if we can find a hyperplane in the n-dimensional vector space R^n, say

$$\alpha 0 + \alpha 1 x1 + \alpha 2 x2 + \cdots + \alpha n xn = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

having the following two properties:

1. For each instance $\vec{x}$ with class label −1 we have

$$\alpha 0 + \alpha 1 x1 + \alpha 2 x2 + \cdots + \alpha n xn < 0.$$

2. For each instance $\vec{x}$ with class label +1 we have

$$\alpha 0 + \alpha 1x1 + \alpha 2x2 + \cdots + \alpha nxn > 0.$$

A hyperplane given by Eq.(1) having the two properties given above is called a separating hyperplane for the data set. If a data set with two class labels is linearly separable, then, in general, there will be several separating hyperplanes for the data set. This is illustrated in the example below.

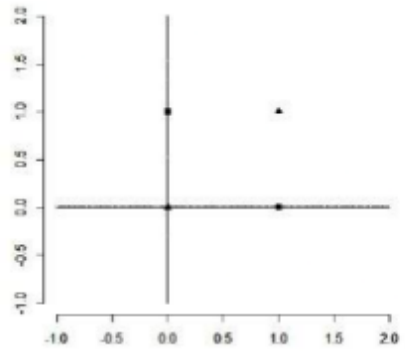
Example:

Show that the dataset in the given table is not separable

X	Y	Class Label
0	0	0
0	1	1
1	0	1
1	1	0

Solution:

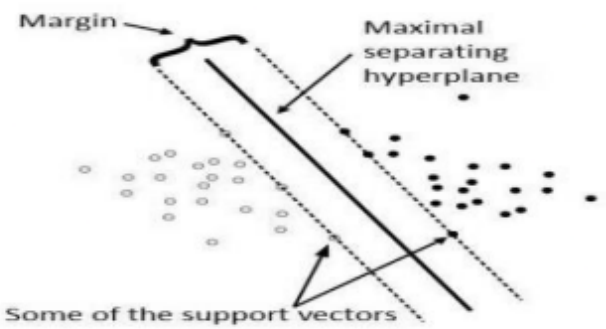
The scatterplot of data in Table shown in Figure below shows that the data is not linearly separable.



Maximal margin hyperplanes

Consider a linearly separable data set having two class labels "−1" and "+1". Consider a separating hyperplane H for the data set.

1. Consider the perpendicular distances from the training instances to the separating hyperplane H and consider the smallest such perpendicular distance. The double of this smallest distance is called the margin of the separating hyperplane H.
2. The hyperplane for which the margin is the largest is called the maximal margin hyperplane (also called maximum margin hyperplane) or the optimal separating hyperplane.
3. The maximal margin hyperplane is also called the support vector machine for the data set.
4. The data points that lie closest to the maximal margin hyperplane are called the support vectors.



Finding maximum margin:

Finding the maximum margin in a Support Vector Machine (SVM) involves training the SVM model to optimize the margin while correctly classifying the data points.

1. Data Preparation

2. Feature Scaling:

Normalize or standardize your feature values. SVM is sensitive to the scale of features, so scaling is essential to ensure that all features have equal importance in defining the margin.

3. Kernel Selection

4. Training the SVM

During training, the SVM algorithm aims to find the optimal hyperplane that maximizes the margin while correctly classifying the data points. The main hyperparameter to adjust is 'C,' which controls the trade-off between margin width and classification error:

- Smaller 'C' values emphasize a larger margin but may tolerate some misclassification.
- Larger 'C' values prioritize correct classification at the cost of a narrower margin.

The optimization problem involves finding the weight vector 'w' and bias term 'b' that define the hyperplane. This is typically done using convex optimization techniques.

5. Margin Calculation

Once the SVM is trained, you can calculate the margin (M). The margin is the perpendicular distance between the hyperplane and the nearest data point (support vector) from any class. We can calculate it using the formula:

$$M = 2 / ||w||$$

Here, 'w' is the weight vector of the hyperplane. The magnitude of 'w' ($||w||$) is inversely proportional to the width of the margin.

6. Support Vector Identification

Identify the support vectors. These are the data points closest to the decision boundary (hyperplane). Support vectors play a crucial role in determining the margin and the position of the decision boundary.

7. Visualization:

To visualize the maximum margin and decision boundary, we can plot the data along with the hyperplane and margin. Support vectors can be marked differently to highlight their significance.

8. Fine-Tuning:

Depending on the problem and the results, we may need to fine-tune the model by adjusting hyperparameters or considering alternative kernel functions. The choice of kernel and 'C' can affect the margin and model performance.

SVM Numerical: Please refer to your class notes

UNIT 3

BAYESIAN AND INSTANCE BASED LEARNING

Marginal Probability:

- Marginal probability is the probability of an event irrespective of the outcome of another variable. e.g. P(A)
- the probability of an event occurring (p(A)), it may be thought of as an unconditional probability. It is not conditioned on another event.
- Example: the probability that a card drawn is red (p(red) = 0.5). Another example: the probability that a card drawn is a 4 (p(four)=1/13).
- The marginal probability is the probability of the event summed over all outcomes for the dependent variables, called the sum rule.

Joint Probability

- Joint probability is the probability of two events occurring simultaneously.
- p(A and B) or P(A, B). The probability of event A and event B occurring.
- It is the probability of the intersection of two or more events. The probability of the intersection of A and B may be written p(A ∩ B).
- Example: the probability that a card is a four and red =p(four and red) = 2/52=1/26. (There are two red fours in a deck of 52, the 4 of hearts and the 4 of diamonds).
- The joint probability is the probability of two (or more) simultaneous events, often described in terms of events A and B from two dependent random variables, e.g. X and Y.
- The joint probability can be calculated using the conditional probability; for example:

  P(A, B) = P(A | B) * P(B)
- This is called the product rule. Importantly, the joint probability is symmetrical, meaning that:

  P(A, B) = P(B, A)
- The conditional probability is the probability of one event given the occurrence of another event, often described in terms of events A and B from two dependent random variables e.g. X and Y.
- Probability of one (or more) event given the occurrence of another event, e.g.

  P(A given B) or P(A | B).
- p(A|B) is the probability of event A occurring, given that event B occurs.
- Example: given that you drew a red card, what's the probability that it's a four (p(four|red))=2/26=1/13. So out of the 26 red cards (given a red card), there are two fours so 2/26=1/13.
- The conditional probability can be calculated using the joint probability; for example:

  P(A | B) = P(A, B) / P(B)
- The conditional probability is not symmetrical; for example:

  P(A | B) != P(B | A)

Bayes' Rule (Theorem) **:**

- Learning and classification methods based on probability theory.
- Baye's theorem plays a critical role in probabilistic learning and classification.
- Uses prior probability of each category given no information about an item.
- Categorization produces a posterior probability distribution over the possible categories given a description of an item.

Let A and B any two events in a random experiment. If P(A) ≠ 0, then

$$P(b|a) = \frac{P(a|b) * P(b)}{P(a)}$$

Proof of bays rule:

We know that:

P(a|b) = P(a ^ b)/ P(b)

P(a ^ b) = P(a|b) P(b).....(1)

Similarly

P(b|a) = P(a ^ b)/ P(a)

P(a ^ b) = P(b|a) P(a) .....(2)

Equating 1 and 2

P(a|b) P(b) = P(b|a) P(a)

i.e. P(b|a) = P(a|b) P(b)/P(a)

Note:-

- The importance of the result is that it helps us to "invert" conditional probabilities, that is, to express the conditional probability P(A|B) in terms of the conditional probability P(B|A).
- The following terminology is used in this context:
  - A is called the proposition and B is called the evidence.
  - P(A) is called the prior probability of proposition and P(B) is called the prior probability of evidence.
  - P(A|B) is called the posterior probability of A given B.
  - P(B|A) is called the likelihood of B given A.

Bayes Formula

$$P(C_i \mid D) = \frac{\text{JointProbability}(C_i \quad \& \quad D)}{\text{MarginalProbability}(D)}$$

$$P(C_i \mid D) = \frac{Likelihood \times Prior}{Total\_evidence}$$

$$P(C_i \mid D) = \frac{P(D \mid C_i)P(C_i)}{P(D)}$$

Thomas Bayes
1702-1761

Generalisation

Let the sample space be divided into disjoint events B1, B2, . . . , Bn and A be any event. Then we have

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

Problem

Consider a set of patients coming for treatment in a certain clinic. Let A denote the event that a "Patient has liver disease" and B the event that a "Patient is an alcoholic." It is known from experience that 10% of the patients entering the clinic have liver disease and 5% of the patients are alcoholics. Also, among those patients diagnosed with liver disease, 7% are alcoholics. Given that a patient is alcoholic, what is the probability that he will have liver disease?

Solution

Using the notations of probability, we have

$P(A) = 10\% = 0.10$

$P(B) = 5\% = 0.05$

$P(B|A) = 7\% = 0.07$

$$P(A|B) = \frac{0.07 \times 0.10}{0.05} = 0.14$$

Why is the Bayes' rule is useful in practice?

Bayes' rule is useful in practice because there are many cases where we have good probability estimates for three of the four probabilities involved, and therefore can compute the fourth one.

Useful for assessing diagnostic probability from causal probability:

$$P(Cause|Effect) = \frac{P(Effect|Cause)P(Cause)}{P(Effect)}$$

Diagnostic knowledge is often more fragile than causal knowledge.

Uses of Bayes' Theorem :

In doing an expert task, such as medical diagnosis, the goal is to determine identifications (diseases) given observations (symptoms). Bayes' Theorem provides such a relationship.

*P(A | B) = P(B | A) * P(A) / P(B)*

Suppose: *A* = Patient has measles, *B* = has a rash

Then: *P(measles/rash) = P(rash/measles) * P(measles) / P(rash)*

The desired diagnostic relationship on the left can be calculated based on the known statistical quantities on the right.

Naive Bayes algorithm

The naive Bayes algorithm is based on the following assumptions:

- All the features are independent and are unrelated to each other. Presence or absence of a feature does not influence the presence or absence of any other feature.

- The data has class-conditional independence, which means that events are independent so long as they are conditioned on the same class value.

These assumptions are, in general, true in many real world problems. It is because of these assumptions, the algorithm is called a naive algorithm.

Basic idea

Suppose we have a training data set consisting of N examples having n features. Let the features be named as (F1, . . . , Fn). A feature vector is of the form (f1, f2, . . . , fn). Associated with each example, there is a certain class label. Let the set of class labels be {c1, c2, . . . , cp}.

Suppose we are given a test instance having the feature vector

X = (x1, x2, . . . , xn)…………………………(2)

We are required to determine the most appropriate class label that should be assigned to the test instance. For this purpose we compute the following conditional probabilities

P(c1|X), P(c2|X), . . . , P(cp|X).

and choose the maximum among them. Let the maximum probability be P(ci|X). Then, we choose ci as the most appropriate class label for the training instance having X as the feature vector. The direct computation of the probabilities given in Eq.(2) are difficult for a number of reasons. The Bayes' theorem can b applied to obtain a simpler method

Computation of probabilities

Using Bayes' theorem, we have:

$$P(ck|X) = \frac{P(X|ck)P(ck)}{\sum_{i=1}^{n} P(X)} \text{………………………………….(3)}$$

Since, by assumption, the data has class-conditional independence, we note that the events "x1|ck", "x2|ck", ⋯, xn|ck are independent (because they are all conditioned on the same class label ck).

Hence we have

$$P(X|ck) = P((x1, x2, \ldots, xn)|ck)$$

$$= P(x1|ck)P(x2|ck)\cdots P(xn|ck)$$

Since the denominator P(X) is independent of the class labels, we have

$$P(ck|X) \propto P(x1|ck)P(x2|ck)\cdots P(xn|ck)P(ck).$$

So it is enough to find the maximum among the following values:

$$P(x1|ck)P(x2|ck)\cdots P(xn|ck)P(ck),\ k = 1, \ldots, p.$$

Note:-

The various probabilities in the above expression are computed as follows:

$$P(ck) = \frac{\text{No.of examples with class label ck}}{\textit{Total Number of Examples}}$$

$$P(xj \mid ck) = \frac{\text{No.of examples with jth feature equal to xj and class label ck}}{\text{No.of examples with class label ck}}$$

Algorithm: Naive Bayes

Let there be a training data set having n features F1, . . . , Fn. Let f1 denote an arbitrary value of F1, f2 of F2, and so on. Let the set of class labels be {c1, c2, . . . , cp}. Let there be given a test instance having the feature vector

X = (x1, x2, . . . , xn).

We are required to determine the most appropriate class label that should be assigned to the test instance.

Step 1. Compute the probabilities P(ck) for k = 1, . . . , p.

Step 2. Form a table showing the conditional probabilities P(f1|ck), P(f2|ck), . . . , P(fn|ck) for all values of f1, f2, . . . , fn and for k = 1, . . . , p.

Step 3. Compute the products qk = P(x1|ck)P(x2|ck)···P(xn|ck)P(ck) for k = 1, . . . , p.

Step 4. Find j such qj = max{q1, q2, . . . , qp}.

Step 5. Assign the class label cj to the test instance X.

Problem

Consider a training data set consisting of the fauna of the world. Each unit has three features named "Swim", "Fly" and "Crawl". Let the possible values of these features be as follows:

Swim Fast, Slow, No

Fly Long, Short, Rarely, No

Crawl Yes, No

For simplicity, each unit is classified as "Animal", "Bird" or "Fish". Let the training data set be as in Table Use naive Bayes algorithm to classify a particular species if its features are (Slow, Rarely, No)

Sl. No.	Swim	Fly	Crawl	Class
1	Fast	No	No	Fish
2	Fast	No	Yes	Animal
3	Slow	No	No	Animal
4	Fast	No	No	Animal
5	No	Short	No	Bird
6	No	Short	No	Bird
7	No	Rarely	No	Animal
8	Slow	No	Yes	Animal
9	Slow	No	No	Fish
10	Slow	No	Yes	Fish
11	No	Long	No	Bird
12	Fast	No	No	Bird

Solution:

In this example, the features are

F1 = "Swim", F2 = "Fly", F3 = "Crawl".

The class labels are

c1 = "Animal", c2 = " Bird", c3 = "Fish".

The test instance is (Slow, Rarely, No) and so we have:

x1 = "Slow", x2 = "Rarely", x3 = "No".

We construct the frequency table shown in Table below which summarizes the data. (It may be noted that the construction of the frequency table is not part of the algorithm.)

Class	Features									Total
	Swim (F_1)			Fly (F_2)				Crawl (F_3)		
	Fast	Slow	No	Long	Short	Rarely	No	Yes	No	
Animal (c_1)	2	2	1	0	0	1	4	2	3	5
Bird (c_2)	1	0	3	1	2	0	1	1	3	4
Fish (c_3)	1	2	0	0	0	0	3	0	3	3
Total	4	4	4	1	2	1	8	4	8	12

Step 1: We compute following probabilities.

$$P(c1) = \frac{\text{No.of records with class label "Animal"}}{\textit{Total Number of Examples}} = 5/12$$

$$P(c2) = \frac{\text{No.of records with class label "Bird"}}{\textit{Total Number of Examples}} = 4/12$$

$$P(c3) = \frac{\text{No.of records with class label "Fish"}}{\textit{Total Number of Examples}} = 3/12$$

Step 2. We construct the following table of conditional probabilities:

Class	Features								
	Swim (F_1) f_1			Fly (F_2) f_2				Crawl (F_3) f_3	
	Fast	Slow	No	Long	Short	Rarely	No	Yes	No
Animal (c_1)	2/5	2/5	1/5	0/5	0/5	1/5	4/5	2/5	3/5
Bird (c_2)	1/4	0/4	3/4	1/4	2/4	0/4	1/4	0/4	4/4
Fish (c_3)	1 3	2/3	0/3	0/3	0/3	0/3	3/3	0/3	3/3

Step 3. We now calculate the following numbers:

$q1 = P(x1|c1)P(x2|c1)P(x3|c1)P(c1)$

$= (2/5) \times (1/5) \times (3/5) \times (5/12) \quad = 0.02$

$q2 = P(x1|c2)P(x2|c2)P(x3|c2)P(c2)$

$= (0/4) \times (0/4) \times (3/4) \times (4/12) \quad = 0$

$q3 = P(x1|c3)P(x2|c3)P(x3|c3)P(c3)$

$= (2/3) \times (0/3) \times (3/3) \times (3/12) \quad = 0$

Step 4. Now max {q1, q2, q3} = 0.05.

Step 5. The maximum is q1 an it corresponds to the class label c1 = " Animal".

So we assign the class label "Animal" to the test instance "(Slow, Rarely, No)".

Maximum likelihood estimation (ML estimation)

To develop a Bayesian classifier, we need the probabilities P(x|ck) for the class labels c1, . . . , ck. These probabilities are estimated from the given data. There is need to know whether the sample is truly random so that the computed probabilities are good approximations to true probabilities. If they are good approximations of true probabilities, then there would be an underlying probability distribution. Suppose we have reasons to believe that the underlying distribution has a particular form, say binomial, Poisson or normal. These forms are defined by probability functions or probability density functions. There are parameters which define these functions, and these parameters are to be estimated to test whether a given data follow some particular distribution. Maximum likelihood estimation is particular method to estimate the parameters of a probability distribution.

Definition

Maximum likelihood estimation (MLE) is a method of estimating the parameters of a statistical model, given observations. MLE attempts to find the parameter values that maximize the likelihood function, given the observations. The resulting estimate is called a maximum likelihood estimate, which is also abbreviated as MLE.

The general MLE method

Suppose we have a random sample $X = \{x1, \ldots, xn\}$ taken from a probability distribution having the probability mass function or probability density function $p(x|\theta)$ where x denotes a value of the random variable and θ denotes the set of parameters that appear in the function. The likelihood of sample X is a function of the parameter θ and is defined as $l(\theta) = p(x1|\theta)p(x2|\theta) \ldots p(xn|\theta)$.

In maximum likelihood estimation, we find the value of θ that makes the value of the likelihood function maximum. For computation convenience, we define the log likelihood function as the logarithm of the likelihood function:

$$L(\theta) = \log l(\theta)$$
$$= \log p(x1|\theta) + \log p(x2|\theta) + \cdots + \log p(xn|\theta).$$

A value of θ that maximizes $L(\theta)$ will also maximise $l(\theta)$ and vice-versa. Hence, in maximum likelihood estimation, we find θ that maximizes the log likelihood function. Sometimes the maximum likelihood estimate of θ is denoted by $\hat{\theta}$.

***Bayesian Belief networks*:**

- *A data structure to represent the dependencies among variables and to give a concise specification of any full joint probability distribution.*
- Also called *belief networks* or *probabilistic network* or *casual network* or *knowledge map*.

The basic idea is:

- Knowledge in the world is *modular* -- most events are conditionally independent of most other events.
- Adopt a model that can use a more local representation to allow interactions between events that *only* affect each other.
- Some events may only be *unidirectional* others may be *bidirectional* -- make a distinction between these in model.
- Events may be causal and thus get chained together in a network.

A Bayesian network is a directed acyclic graph which consists of:

A set of random variables which makes up the nodes of the network.

- A set of directed links (arrows) connecting pairs of nodes. If there is an arrow from node X to node Y, X is said to be a parent of Y.
- Each node Xi has a conditional probability distribution P(Xi| Parents(Xi)) that quantifies the effect of the parents on the node.

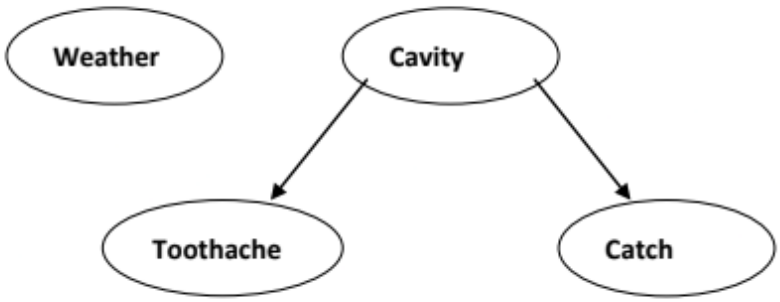
Intuitions:

- A Bayesian network models our incomplete understanding of the causal relationships from an application domain.
- A node represents some state of affairs or event.
- A link from X to Y means that X has a direct influence on Y.

Implementation:

- A *Bayesian Network* is a *directed acyclic graph*:
  - A graph where the directions are links which indicate dependencies that exist between nodes.
  - Nodes represent propositions about events or events themselves.
  - Conditional probabilities quantify the strength of dependencies.

Our existing simple world of variables *toothache, cavity, catch* & *weather* is represented as:

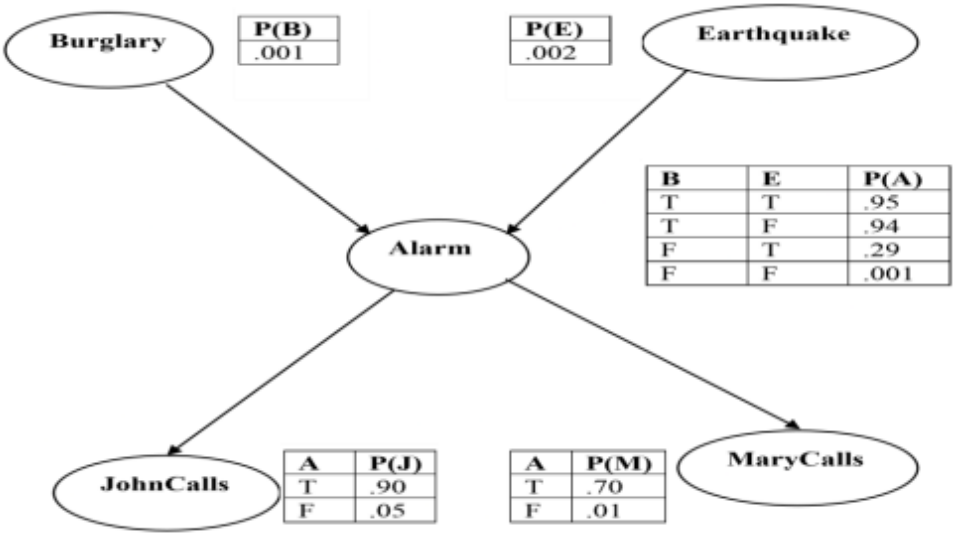


- *Weather* is independent of the other variables

Example: Sample Domain:

You have a burglar alarm installed in your home. It is fairly reliable at detecting a burglary, but also responds on occasion to minor earthquakes. You also have two neighbors, John and Mary, who have promised to call you at work when they hear the alarm. John always calls when he hears the alarm, but sometimes confuses the telephone ringing with the alarm and calls then, too. Mary, on the other hand, likes rather loud music and sometimes misses the alarm altogether.

- We would like have to estimate the probability of a burglary with given evidence who has or has not call.
- **Variables:**Burglary, Earthquake, Alarm, JohnCalls, MaryCalls



The probabilities associated with the nodes reflect our representation of the causal relationships.

A Bayesian network provides a complete description of the domain in the sense that one can compute the probability of any state of the world (represented as a particular assignment to each variable).

Example: What is the probability that the alarm has sounded, but neitherburglary nor an earthquake has occurred, and both John and Mary call?

$P(j, m, a, \neg b, \neg e) = P(j|a)\ P(m|a)\ P(a|, \neg b, \neg e)\ P(\neg b)\ P(\neg e)$

$= 0.90*0.70*0.001*0.999*0.998 = 0.00062$

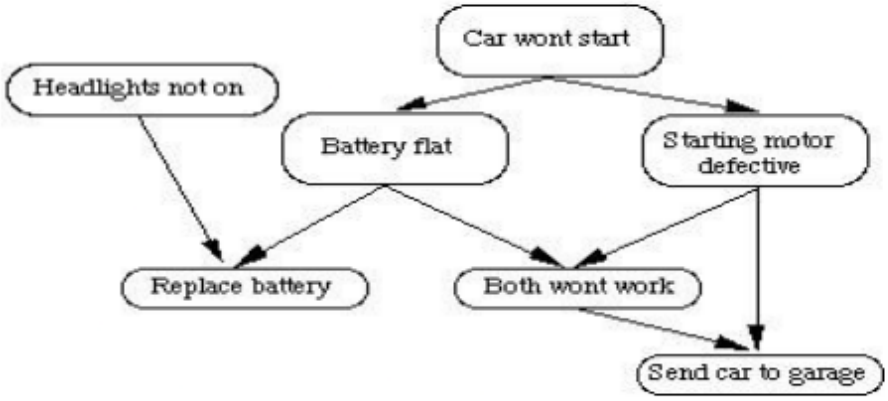
Consider the following example:

- The probability, $P(S_1)$ that my car won't start.
- If my car won't start then it is likely that
- The battery is flat or
- The staring motor is broken.

In order to decide whether to fix the car myself or send it to the garage I make the following decision:

- If the headlights do not work then the battery is likely to be flat so i fix it myself.
- If the starting motor is defective then send car to garage.
- If battery and starting motor both gone send car to garage.

The Bayesian network to represent this is as follows:



***K-Nearest Neighbor(KNN) Algorithm*:**

K-Nearest Neighbours is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining, and intrusion detection. 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.

It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data). We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

As an example, consider the following table of data points containing two features:

Now, given another set of data points (also called testing data), allocate these points to a group by analyzing the training set. Note that the unclassified points are marked as 'White'.

Intuition Behind KNN Algorithm

If we plot these points on a graph, we may be able to locate some clusters or groups. Now, given an unclassified point, we can assign it to a group by observing what group its nearest neighbors belong to. This means a point close to a cluster of points classified as 'Red' has a higher probability of getting classified as 'Red'.

Intuitively, we can see that the first point (2.5, 7) should be classified as 'Green' and the second point (5.5, 4.5) should be classified as 'Red'.

Distance Metrics Used in KNN Algorithm

As we know that the KNN algorithm helps us identify the nearest points or the groups for a query point. But to determine the closest groups or the nearest points for a query point we need some metric. For this purpose, we use below distance metrics:

Euclidean Distance

This is nothing but the cartesian distance between the two points which are in the plane/hyperplane. Euclidean distance can also be visualized as the length of the straight line that joins the two points which are into consideration. This metric helps us calculate the net displacement done between the two states of an object.

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

Manhattan Distance

This distance metric is generally used when we are interested in the total distance traveled by the object instead of the displacement. This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions.

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

Minkowski Distance

We can say that the Euclidean, as well as the Manhattan distance, are special cases of the Minkowski distance.

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

From the formula above we can say that when p = 2 then it is the same as the formula for the Euclidean distance and when p = 1 then we obtain the formula for the Manhattan distance.

The above-discussed metrics are most common while dealing with a Machine Learning problem but there are other distance metrics as well like Hamming Distance which come in handy while dealing with problems that require overlapping comparisons between two vectors whose contents can be boolean as well as string values.

How to choose the value of k for KNN Algorithm?

The value of k is very crucial in the KNN algorithm to define the number of neighbors in the algorithm. The value of k in the k-nearest neighbors (k-NN) algorithm should be chosen based on the input data. If the input data has more outliers or noise, a higher value of k would be better. It is recommended to choose an odd value for k to avoid ties in classification. Cross-validation methods can help in selecting the best k value for the given dataset.

Applications of the KNN Algorithm

Data Preprocessing – While dealing with any Machine Learning problem we first perform the EDA part in which if we find that the data contains missing values then there are multiple imputation methods are available as well. One of such method is KNN Imputer which is quite effective ad generally used for sophisticated imputation methodologies.

Pattern Recognition – KNN algorithms work very well if you have trained a KNN algorithm using the MNIST dataset and then performed the evaluation process then you must have come across the fact that the accuracy is too high.

Recommendation Engines – The main task which is performed by a KNN algorithm is to assign a new query point to a pre-existed group that has been created using a huge corpus of datasets. This is exactly what is required in the recommender systems to assign each user to a particular group and then provide them recommendations based on that group's preferences.

Advantages of the KNN Algorithm

Easy to implement as the complexity of the algorithm is not that high.

Adapts Easily – As per the working of the KNN algorithm it stores all the data in memory storage and hence whenever a new example or data point is added then the algorithm adjusts itself as per that new example and has its contribution to the future predictions as well.

Few Hyperparameters – The only parameters which are required in the training of a KNN algorithm are the value of k and the choice of the distance metric which we would like to choose from our evaluation metric.

Disadvantages of the KNN Algorithm

Does not scale – As we have heard about this that the KNN algorithm is also considered a Lazy Algorithm. The main significance of this term is that this takes lots of computing power as well as data storage. This makes this algorithm both time-consuming and resource exhausting.

Curse of Dimensionality – There is a term known as the peaking phenomenon according to this the KNN algorithm is affected by the curse of dimensionality which implies the algorithm faces a hard time classifying the data points properly when the dimensionality is too high.

Prone to Overfitting – As the algorithm is affected due to the curse of dimensionality it is prone to the problem of overfitting as well. Hence generally feature selection as well as dimensionality reduction techniques are applied to deal with this problem.

Algorithm:

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready

Example: Refer to your class notes

UNIT 4

INTRODUCTION TO UN-SUPERVISED LEARNING AND DIMENSIONALITY REDUCTION

Introduction to Clustering

Clustering or cluster analysis is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters). Clustering is a main task of exploratory data mining and used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances between cluster members, dense areas of the data space etc.

Examples of data with natural clusters

In many applications, there will naturally be several groups or clusters in samples.

In speech recognition, where the same word can be uttered in different ways, due to different pronunciation, accent, gender, age, and so forth, there is not a single, universal prototype. In a large sample of utterances of a specific word, All the different ways should be represented in the sample.

K-means clustering

The k-means clustering algorithm is one of the simplest unsupervised learning algorithms for solving the clustering problem.

Let it be required to classify a given data set into a certain number of clusters, say, k clusters. We start by choosing k points arbitrarily as the "centres" of the clusters, one for each cluster. We then associate each of the given data points with the nearest centre. We now take the averages of the data points associated with a centre and replace the centre with the average, and this is done for each of the centres. We repeat the process until the centres converge to some fixed points. The data points nearest to the centres form the various clusters in the dataset. Each cluster is represented by the associated centre. The distance between the points (x1, x2) and (y1, y2) will be calculated using the familiar distance formula of elementary analytical geometry:

$$\sqrt{(x1 - y1)^2 + (x2 - y2)^2}.$$

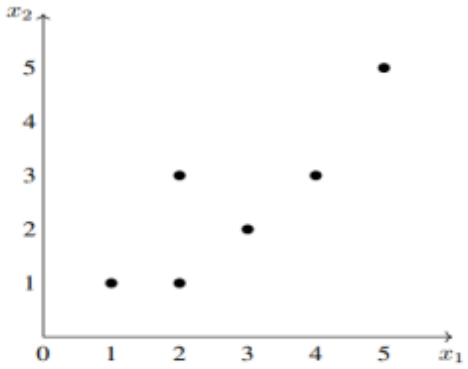
Problem

Use k-means clustering algorithm to divide the following data into two clusters and also compute the representative data points for the clusters.

x1 1 2 2 3 4 5

x2 1 1 3 2 3 5

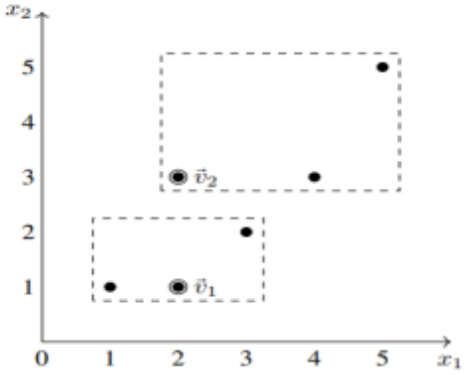
Solution:



1. In the problem, the required number of clusters is 2 and we take k = 2.

2. We choose two points arbitrarily as the initial cluster centres. Let us choose arbitrarily v⃗1 = (2, 1), v⃗2 = (2, 3).

3. We compute the distances of the given data points from the cluster centers



$\vec{x}_i$	Data point	Distance from $\vec{v}_1 = (2,1)$	Distance from $\vec{v}_2 = (2,3)$	Minimum distance	Assigned center
$\vec{x}_1$	$(1,1)$	1	2.24	1	$\vec{v}_1$
$\vec{x}_2$	$(2,1)$	0	2	0	$\vec{v}_1$
$\vec{x}_3$	$(2,3)$	2	0	0	$\vec{v}_2$
$\vec{x}_4$	$(3,2)$	1.41	1.41	0	$\vec{v}_1$
$\vec{x}_5$	$(4,3)$	2.82	2	2	$\vec{v}_2$
$\vec{x}_6$	$(5,5)$	5	3.61	3.61	$\vec{v}_2$

Cluster 1: {x⃗1, x⃗2, x⃗4} represented by v⃗1 Number of data points in Cluster 1: c1 = 3. Cluster 2 : {x⃗3, x⃗5, x⃗6} represented by v⃗2 Number of data points in Cluster 2: c2 = 3.

4. The cluster centres are recalculated as follows:

$$\vec{v}1 = \frac{1}{c1}(\vec{x}1 + \vec{x}2 + \vec{x}4)$$

$$= (2.00, 1.33)$$

$$\vec{v}2 = \frac{1}{c2}(\vec{x}3 + \vec{x}5 + \vec{x}6)$$

$$= (3.67, 3.67)$$

5. We compute the distances of the given data points from the new cluster centers

$\vec{x}_i$	Data point	Distance from $\vec{v}_1 = (2,1)$	Distance from $\vec{v}_2 = (2,3)$	Minimum distance	Assigned center
$\vec{x}_1$	$(1,1)$	1.05	3.77	1.05	$\vec{v}_1$
$\vec{x}_2$	$(2,1)$	0.33	3.14	0.33	$\vec{v}_1$
$\vec{x}_3$	$(2,3)$	1.67	1.80	1.67	$\vec{v}_1$
$\vec{x}_4$	$(3,2)$	1.20	1.80	1.20	$\vec{v}_1$
$\vec{x}_5$	$(4,3)$	2.60	0.75	0.75	$\vec{v}_2$
$\vec{x}_6$	$(5,5)$	4.74	1.89	1.89	$\vec{v}_2$

Cluster 1 : {x⃗1, x⃗2, x⃗3, x⃗4} represented by v⃗1 Number of data points in Cluster 1: c1 = 4.

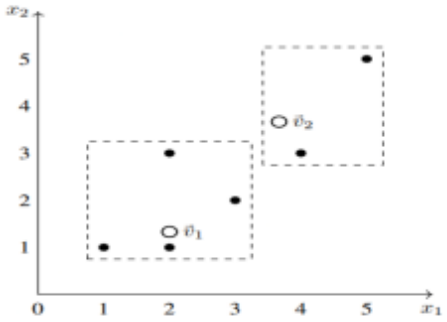
Cluster 2 : {x⃗5, x⃗6} represented by v⃗2 Number of data points in Cluster 1: c2 = 2.

6. The cluster centres are recalculated as follows:

v⃗1 ==$\frac{1}{c1}$(x⃗1 + x⃗2 + x⃗3 + x⃗4)

= (2.00, 1.33)

v⃗2 =$\frac{1}{c2}$(x⃗5 + x⃗6) = (3.67, 3.67



7. We compute the distances of the given data points from the new cluster centers.

4.609772 3.905125 2.692582 2.500000 1.118034 1.118034

$\vec{x}_i$	Data point	Distance from $\vec{v}_1 = (2,1)$	Distance from $\vec{v}_2 = (2,3)$	Minimum distance	Assigned center
$\vec{x}_1$	$(1,1)$	1.25	4.61	1.25	$\vec{v}_1$
$\vec{x}_2$	$(2,1)$	0.75	3.91	0.75	$\vec{v}_1$
$\vec{x}_3$	$(2,3)$	1.25	2.69	1.25	$\vec{v}_1$
$\vec{x}_4$	$(3,2)$	1.03	2.50	1.03	$\vec{v}_1$
$\vec{x}_5$	$(4,3)$	2.36	1.12	1.12	$\vec{v}_2$
$\vec{x}_6$	$(5,5)$	4.42	1.12	1.12	$\vec{v}_2$

Cluster 1 : {x⃗1, x⃗2, x⃗3, x⃗4} represented by v⃗1

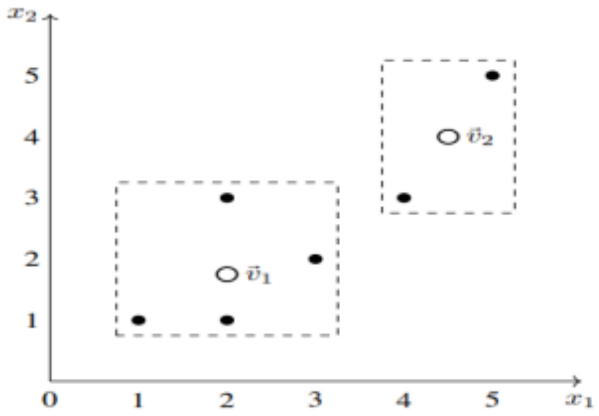
Number of data points in Cluster 1: c1 = 4.

Cluster 2 : {x⃗5, x⃗6} represented by v⃗2

Number of data points in Cluster 1: c1 = 2

8. The cluster centres are recalculated as follows:

$$\begin{aligned}\vec{v}_1 &= \frac{1}{c_1}(\vec{x}_1 + \vec{x}_2 + \vec{x}_3 + \vec{x}_4) \\ &= \frac{1}{4}(\vec{x}_1 + \vec{x}_2 + \vec{x}_3 + \vec{x}_4) \\ &= (2.00, 1.75) \\ \vec{v}_2 &= \frac{1}{c_2}(\vec{x}_5 + \vec{x}_6) \\ &= \frac{1}{2}(\vec{x}_5 + \vec{x}_6) \\ &= (4.00, 4.50)\end{aligned}$$



9. This divides the data into two clusters

Cluster 1 : {x⃗1, x⃗2, x⃗3, x⃗4} represented by v⃗1

Cluster 2 : {x⃗5, x⃗6} represented by v⃗2

10. The cluster centres are recalculated as follows:

v⃗1 = (2.00, 1.75)

v⃗2 = (4.00, 4.50)

We note that these are identical to the cluster centres calculated in Step 8. So there will be no reassignment of data points to different clusters and hence the computations are stopped here.

11. Conclusion: The k means clustering algorithm with k = 2 applied to the dataset in Table 13.1 yields the following clusters and the associated cluster centres:

Cluster 1 : {x⃗1, x⃗2, x⃗3, x⃗4} represented by v⃗1 = (2.00, 1.75) Cluster 2 : {x⃗5, x⃗6} represented by v⃗2 = (2.00, 4.75)

Algorithm

Step 1. Randomly select k cluster centers v⃗1, . . . , v⃗k.

Step 2. Calculate the distance between each data point x⃗i and each cluster center v⃗j .

Step 3. For each j = 1, 2, . . . , N, assign the data point x⃗j to the cluster center v⃗i for which the distance ||x⃗j − v⃗i|| is minimum. Let x⃗i1, x⃗i2, . . ., x⃗ici be the data points assigned to v⃗i

.Step 4. Recalculate the cluster centres using

$$\vec{v}i = \frac{1}{c1}(\vec{x}i1 + \cdots + \vec{x}ici), i = 1, 2, \ldots, k.$$

Step 5. Recalculate the distance between each data point and newly obtained cluster centers.

Step 6. If no data point was reassigned then stop. Otherwise repeat from Step 3.

Some methods for initialisation

The following are some of the methods for choosing the initial vi's.

- Randomly take some k data points as the initial vi's.
- Calculate the mean of all data and add small random vectors to the mean to get the k initial vi's.
- Calculate the principal component, divide its range into k equal intervals, partition the data into k groups, and then take the means of these groups as the initial centres.

Disadvantages

Even though the k-means algorithm is fast, robust and easy to understand, there are several disadvantages to the algorithm.

- The learning algorithm requires apriori specification of the number of cluster centers.
- The final cluster centres depend on the initial vi's.
- With different representation of data we get different results (data represented in form of cartesian co-ordinates and polar co-ordinates will give different results).
- Euclidean distance measures can unequally weight underlying factors.
- The learning algorithm provides the local optima of the squared error function.
- Randomly choosing of the initial cluster centres may not lead to a fruitful result.
- The algorithm cannot be applied to categorical data.

Application: Image segmentation and compression

Image segmentation

The goal of segmentation is to partition an image into regions each of which has a reasonably homogeneous visual appearance or which corresponds to objects or parts of objects. Each pixel in an image is a point in a 3-dimensional space comprising the intensities of the red, blue, and green channels. A segmentation algorithm simply treats each pixel in the image as a separate data point.

For any value of k, each pixel is replaced by the pixel vector with the (R, G, B) intensity triplet given by the centre μk to which that pixel has been assigned. For a given value of k, the algorithm is representing the image using a palette of only k colours. It should be emphasized that this use of

k-means is a very crude approach to image segmentation. The image segmentation problem is in general extremely difficult.

Data compression

We can also the clustering algorithm to perform data compression. There are two types of data compression: lossless data compression, in which the goal is to be able to reconstruct the original data exactly from the compressed representation, and lossy data compression, in which we accept some errors in the reconstruction in return for higher levels of compression than can be achieved in the lossless case.

We can apply the k-means algorithm to the problem of lossy data compression as follows. For each of the N data points, we store only the identity of the cluster to which it is assigned. We also store the values of the k cluster centres μk, which requires much less data, provided we choose k much smaller than N. Each data point is then approximated by its nearest centre μk. New data points can similarly be compressed by first finding the nearest μk and then storing the label k instead of the original data vector. This framework is often called vector quantization, and the vectors Îijμk are called code-book vectors.

Hierarchical clustering

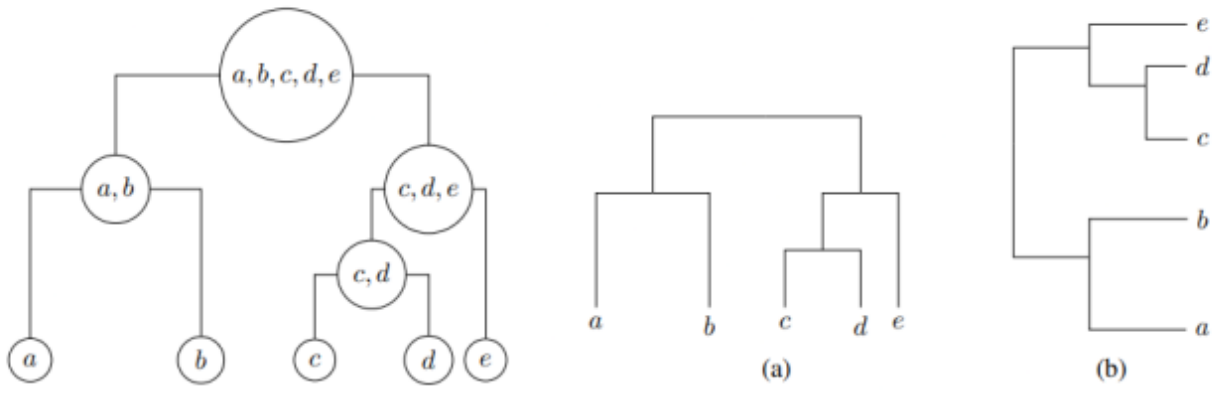
Hierarchical clustering (also called hierarchical cluster analysis or HCA) is a method of cluster analysis which seeks to build a hierarchy of clusters (or groups) in a given dataset. The hierarchical clustering produces clusters in which the clusters at each level of the hierarchy are created by merging 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.

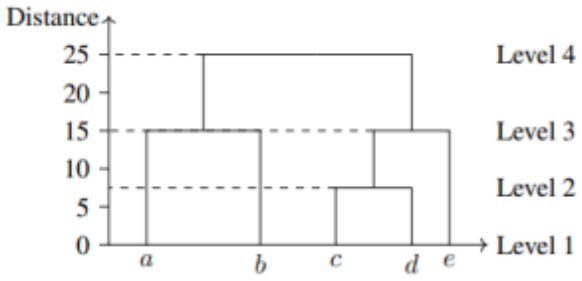
Dendrograms

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. 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 . It may also be drawn with the root node at the left and the branches growing horizontally rightwards . 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 below 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 below shows the simplified format of the dendrogram in Figure . Figure 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.





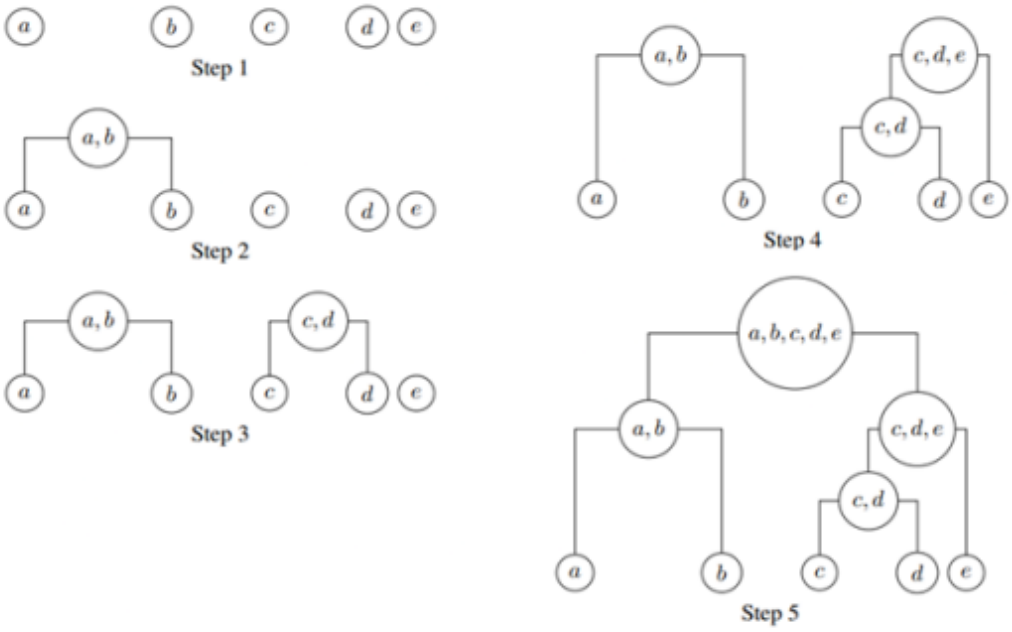
Methods for hierarchical clustering

There are two methods for the hierarchical clustering of a dataset. These are known as the agglomerative method (or the bottom-up method) and the divisive method (or, the top-down method)

Agglomerative method

In the agglomerative we start at the bottom and at each level recursively merge a selected pair of clusters into a single cluster. This produces a grouping at the next higher level with one less cluster. If there are N observations in the dataset, there will be N −1 levels in the hierarchy. The pair chosen for merging consist of the two groups with the smallest "intergroup dissimilarity".

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



Dimensionality Reduction Technique:

In many learning problems, the datasets have large number of variables. Sometimes, the number of variables is more than the number of observations. For example, such situations have arisen in many scientific fields such as image processing, mass spectrometry, time series analysis, internet search engines, and automatic text analysis among others. Statistical and machine learning methods have some difficulty when dealing with such

high-dimensional data. Normally the number of input variables is reduced before the machine learning algorithms can be successfully applied.

In statistical and machine learning, dimensionality reduction or dimension reduction is the process of reducing the number of variables under consideration by obtaining a smaller set of principal variables.

Dimensionality reduction may be implemented in two ways.

• Feature selection

In feature selection, we are interested in finding k of the total of n features that give us the most information and we discard the other (n−k) dimensions. We are going to discuss subset selection as a feature selection method.

• Feature extraction

In feature extraction, we are interested in finding a new set of k features that are the combination of the original n features. These methods may be supervised or unsupervised depending on whether or not they use the output information. The best known and most widely used feature extraction methods are Principal Components Analysis (PCA) and Linear Discriminant Analysis (LDA), which are both linear projection methods, unsupervised and supervised respectively.

Why dimensionality reduction is useful ?

Ans: There are several reasons why we are interested in reducing dimensionality.

- In most learning algorithms, the complexity depends on the number of input dimensions, d, as well as on the size of the data sample, N, and for reduced memory and computation, we are interested in reducing the dimensionality of the problem. Decreasing d also decreases the complexity of the inference algorithm during testing.
- When an input is decided to be unnecessary, we save the cost of extracting it.
- Simpler models are more robust on small datasets. Simpler models have less variance, that is, they vary less depending on the particulars of a sample, including noise, outliers, and so forth.
- When data can be explained with fewer features, we get a better idea about the process that underlies the data, which allows knowledge extraction.
- When data can be represented in a few dimensions without loss of information, it can be plotted and analyzed visually for structure and outliers

Principal component analysis

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the smaller of the number of original variables or the number of observations. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.

Computation of the principal component vectors(PCA algorithm)

The following is an outline of the procedure for performing a principal component analysis on a given data. The procedure is heavily dependent on mathematical concepts. A knowledge of these concepts is essential to carry out this procedure.

Step 1. Data

We consider a dataset having n features or variables denoted by X1, X2, . . . , Xn. Let there be N examples. Let the values of the i-th feature Xi be Xi1, Xi2, . . . , XiN

Features	Example 1	Example 2	⋯	Example N
X_1	X_{11}	X_{12}	⋯	X_{1N}
X_2	X_{21}	X_{22}	⋯	X_{2N}
⋮				
X_i	X_{i1}	X_{i2}	⋯	X_{iN}
⋮				
X_n	X_{n1}	X_{n2}	⋯	X_{nN}

Step 2. Compute the means of the variables

We compute the mean $\bar{X}_i$ of the variable X_i:

$$\bar{X}_i = \frac{1}{N}(X_{i1} + X_{i2} + \cdots + X_{iN}).$$

Step 3. Calculate the covariance matrix

Consider the variables X_i and X_j (i and j need not be different). The covariance of the ordered pair (X_i, X_j) is defined as[1]

$$\text{Cov}\,(X_i, X_j) = \frac{1}{N-1}\sum_{k=1}^{N}(X_{ik} - \bar{X}_i)(X_{jk} - \bar{X}_j).$$

We calculate the following $n \times n$ matrix S called the covariance matrix of the data. The element in the i-th row j-th column is the covariance $\text{Cov}\,(X_i, X_j)$:

$$S = \begin{bmatrix} \text{Cov}\,(X_1, X_1) & \text{Cov}\,(X_1, X_2) & \cdots & \text{Cov}\,(X_1, X_n) \\ \text{Cov}\,(X_2, X_1) & \text{Cov}\,(X_2, X_2) & \cdots & \text{Cov}\,(X_2, X_n) \\ \vdots & & & \\ \text{Cov}\,(X_n, X_1) & \text{Cov}\,(X_n, X_2) & \cdots & \text{Cov}\,(X_n, X_n) \end{bmatrix}$$

Step 4. Calculate the eigenvalues and eigenvectors of the covariance matrix

Let S be the covariance matrix and let I be the identity matrix having the same dimension as the dimension of S.

i) Set up the equation:

$$\det(S - \lambda I) = 0.$$

This is a polynomial equation of degree n in λ. It has n real roots (some of the roots may be repeated) and these roots are the eigenvalues of S. We find the n roots $\lambda_1, \lambda_2, \ldots, \lambda_n$

ii) If $\lambda = \lambda'$ is an eigenvalue, then the corresponding eigenvector is a vector

$$U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$$

such that

$$(S - \lambda' I)U = 0.$$

(This is a system of n homogeneous linear equations in $u_1, u_2, \ldots, u_n$ and it always has a nontrivial solution.) We next find a set of n orthogonal eigenvectors $U_1, U_2, \ldots, U_n$ such that U_i is an eigenvector corresponding to λ_i.[2]

iii) We now normalise the eigenvectors. Given any vector X we normalise it by dividing X by its length. The length (or, the norm) of the vector

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

is defined as

$$\|X\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

Given any eigenvector U, the corresponding normalised eigenvector is computed as

$$\frac{1}{\|U\|} U.$$

We compute the n normalised eigenvectors $e_1, e_2, \ldots, e_n$ by

$$e_i = \frac{1}{\|U_i\|} U_i, \quad i = 1, 2, \ldots, n.$$

Step 5. Derive new data set

Order the eigenvalues from highest to lowest. The unit eigenvector corresponding to the largest eigenvalue is the first principal component. The unit eigenvector corresponding to the next highest eigenvalue is the second principal component, and so on.

i) Let the eigenvalues in descending order be $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ and let the corresponding unit eigenvectors be $e_1, e_2, \ldots, e_n$.

ii) Choose a positive integer p such that $1 \leq p \leq n$.

iii) Choose the eigenvectors corresponding to the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ and form the following $p \times n$ matrix (we write the eigenvectors as row vectors):

$$F = \begin{bmatrix} e_1^T \\ e_2^T \\ \vdots \\ e_p^T \end{bmatrix},$$

where T in the superscript denotes the transpose.

iv) We form the following $n \times N$ matrix:

$$X = \begin{bmatrix} X_{11} - \bar{X}_1 & X_{12} - \bar{X}_1 & \cdots & X_{1N} - \bar{X}_1 \\ X_{21} - \bar{X}_2 & X_{22} - \bar{X}_2 & \cdots & X_{2N} - \bar{X}_2 \\ \vdots & & & \\ X_{n1} - \bar{X}_n & X_{n2} - \bar{X}_n & \cdots & X_{nN} - \bar{X}_n \end{bmatrix}$$

v) Next compute the matrix:

$$X_{\text{new}} = FX.$$

Note that this is a $p \times N$ matrix. This gives us a dataset of N samples having p features.

Step 6. New dataset

The matrix X_{new} is the new dataset. Each row of this matrix represents the values of a feature. Since there are only p rows, the new dataset has only features.

Step 7. Conclusion

This is how the principal component analysis helps us in dimensional reduction of the dataset. Note that it is not possible to get back the original n-dimensional dataset from the new dataset.

UNIT 5

MEASURES FOR PERFORMANCE EVALUATION OF ML ALGORITHMS

Classification Accuracy

- Classification accuracy is a metric that summarizes the performance of a classification model as the number of correct predictions divided by the total number of predictions

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

- For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

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

TP = True Positives: The number of instances that were correctly classified as positive.

TN = True Negatives: The number of instances that were correctly classified as negative.

FP = False Positives: The number of instances that were incorrectly classified as positive.

FN = False Negatives: The number of instances that were incorrectly classified as negative.

- Conversely, the error rate can be calculated as the total number of incorrect predictions made on the test set divided by all predictions made on the test set.

  Error Rate = Incorrect Predictions / Total Predictions

- The accuracy and error rate are complements of each other, meaning that we can always calculate one from the other. For example:

Accuracy = 1 – Error Rate

Error Rate = 1 – Accuracy

Measuring error :- True positive, false positive, etc.

Definitions:- Consider a binary classification model derived from a two-class dataset. Let the class labels be c and ¬c. Let x be a test instance.

1. **True positive:-** Let the true class label of x be c. If the model predicts the class label of x as c, then we say that the classification of x is true positive.
2. **False negative:-** Let the true class label of x be c. If the model predicts the class label of x as ¬c, then we say that the classification of x is false negative.
3. **True negative:-** Let the true class label of x be ¬c. If the model predicts the class label of x as ¬c, then we say that the classification of x is true negative.
4. **False positive:-** Let the true class label of x be ¬c. If the model predicts the class label of x as c, then we say that the classification of x is false positive.

	Actual label of x is c	Actual label of x is $\neg c$
Predicted label of x is c	True positive	False positive
Predicted label of x is $\neg c$	False negative	True negative

Confusion matrix

A confusion matrix is used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. A confusion matrix is a table that categorizes predictions according to whether they match the actual value.

Two-class datasets

For a two-class dataset, a confusion matrix is a table with two rows and two columns that reports the number of false positives, false negatives, true positives, and true negatives. Assume that a classifier is applied to a two-class test dataset for which the true values are known. Let TP denote the number of true positives in the predicted values, TN the number of true negatives, etc. Then the confusion matrix of the predicted values can be represented as follows:

	Actual condition is true	Actual condition is false
Predicted condition is true	TP	FP
Predicted condition is false	FN	FN

Precision and recall

In machine learning, precision and recall are two measures used to assess the quality of results produced by a binary classifier. They are formally defined as follows.

Definitions

Let a binary classifier classify a collection of test data. Let

TP = Number of true positives

TN = Number of true negatives

FP = Number of false positives

FN = Number of false negatives

The precision P is defined as $P = \frac{TP}{TP+FP}$

The recall R is defined as $R = \frac{TP}{TP+FN}$

Problem

Suppose a computer program for recognizing dogs in photographs identifies eight dogs in a picture containing 12 dogs and some cats. Of the eight dogs identified, five actually are dogs while the rest are cats. Compute the precision and recall of the computer program.

Solution

We have: TP = 5 , FP = 3 , FN = 7

The precision P is $P = \frac{TP}{TP+FP} = \frac{5}{5+3} = \frac{5}{8}$

The recall R is $R = \frac{TP}{TP+FN} = \frac{5}{5+7} = \frac{5}{12}$

Other measures of performance

Using the data in the confusion matrix of a classifier of two-class dataset, several measures of performance have been defined. A few of them are listed below.

1. Accuracy = $\frac{TP+TN}{TP+FN+FP+FN}$
2. Error rate = 1− Accuracy
3. Sensitivity = $\frac{TP}{TP+FN}$
4. Specificity = $\frac{TN}{TN+FP}$
5. F-measure = $\frac{2*TP}{2*TP+FP+FN}$

Misclassification costs

- Misclassification costs are the penalties associated with making incorrect predictions in a classification problem.
- In binary classification, there are two possible outcomes: true positive (TP) and true negative (TN), or false positive (FP) and false negative (FN).
- The misclassification costs are typically represented in a cost matrix, where each cell represents the cost of making a particular type of error.
- Confusion Matrix forms the basis for the other types of metrics.
- The error rate can also be calculated from the confusion matrix as the sum of incorrect cells of the table (false positives and false negatives) divided by all cells of the table.
  Error Rate = (FP + FN) / (TP + FN + FP + TN)

Metrics for Evaluating Classifiers

- **True positives (TP):** These refer to the positive tuples that were correctly labeled by the classifier. Let TP be the number of true positives.
- **True negatives(TN):** These are the negative tuples that were correctly labeled by the classifier. Let TN be the number of true negatives.
- **False positives (FP):** These are the negative tuples that were incorrectly labeled as positive (e.g., tuples of class buys computer = no for which the classifier predicted buys computer = yes). Let FP be the number of false positives.
- **False negatives (FN):** These are the positive tuples that were mislabeled as negative (e.g., tuples of class buys computer = yes for which the classifier predicted buys computer = no). Let FN be the number of false negatives.
- Confusion matrix:

		Predicted class		
		yes	*no*	Total
Actual class	*yes*	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	P'	N'	$P + N$

Confusion matrix, shown with totals for positive and negative tuples.

Precision, Recall, F-Measure

- **Precision**: measures the proportion of true positive predictions out of all positive predictions made by the model. It answers the question "What fraction of positive predictions are correct?"
- **Recall**: measures the proportion of true positive predictions out of all actual positive cases. It answers the question "What fraction of positive cases did the model identify correctly?"
- **F-Score** (also known as F1-Score): is the harmonic mean of precision and recall. It provides a single value that balances precision and recall, and is particularly useful when there is an imbalance between the positive and negative classes in the data.
- In summary, precision measures the accuracy of positive predictions, recall measures the completeness of positive predictions, and F-Score balances both to provide a comprehensive evaluation of a binary classifier's performance.

- Precision, Recall, F-Measure
  - Used in information retrieval system
  - Precision(P) = (TP)/(TP+FP)
    - Measure of exactness (i.e. , what percentage of tuples labeled as positive are actually such).
  - Recall(R) = (TP)/(TP+FN)=TP/P
    - Measure of completeness (what percentage of positive tuples are labeled as such).
  - F-measure = (2* Precision * Recall) / (Precision + Recall)
    - Harmonic mean of precision and recall

Find out the precision, recall on following data:

True Positives (TP): 8	False Positives (FP): 2
False Negatives (FN): 3	True Negatives (TN): 17

Precision measures the percentage of **emails flagged as spam** that were correctly classified—that is, the percentage of dots to the right of the threshold line that are green in Figure 1:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{8}{8 + 2} = 0.8$$

Recall measures the percentage of **actual spam emails** that were correctly classified—that is, the percentage of green dots that are to the right of the threshold line in Figure 1:

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{8}{8 + 3} = 0.73$$

Sensitivity and specificity

- Sensitivity and specificity are two measures of the accuracy of a diagnostic test.
- They are often used together to give a more complete picture of the test's performance.
- **Sensitivity**
  - Sensitivity is the ability of a test to correctly identify people with the disease.

- It is calculated as the number of true positives divided by the sum of the true positives and false negatives.
- The sensitivity of a test (also called the true positive rate) is defined as the proportion of people with the disease who will have a positive result
- In other words, a highly sensitive test is one that correctly identifies patients with a disease. A test that is 100% sensitive will identify all patients who have the disease.

- **Specificity**
  - Specificity is the ability of a test to correctly identify people without the disease.
  - It is calculated as the number of true negatives divided by the sum of the true negatives and false positives.
  - The specificity of a test (also called the True Negative Rate) is the proportion of people without the disease who will have a negative result.
  - In other words, the specificity of a test refers to how well a test identifies patients who do not have a disease. A test that has 100% specificity will identify 100% of patients who do not have the disease.
  - A perfect test would have 100% sensitivity and 100% specificity. However, in practice, no test is perfect. There will always be some false positives and false negatives.

Measure	Definition	Formula
Sensitivity	Ability to correctly identify people with the disease	TP / (TP + FN)
Specificity	Ability to correctly identify people without the disease	TN / (TN + FP)

Receiver Operating Characteristic (ROC)

The acronym ROC stands for Receiver Operating Characteristic, a terminology coming from signal detection theory. The ROC curve was first developed by electrical engineers and radar engineers during World War II for detecting enemy objects in battlefields. They are now increasingly used in machine learning and data mining research.

TPR and FPR

Let a binary classifier classify a collection of test data. Let, as before,

TP = Number of true positives

TN = Number of true negatives

FP = Number of false positives

FN = Number of false negatives

Now we introduce the following terminology:

TPR = True Positive Rate = $\frac{TP}{TP+FN}$

= Fraction of positive examples correctly classified

= Sensitivity

FPR = False Positive Rate = $\frac{FP}{FP+TN}$

= Fraction of negative examples incorrectly classified

= 1 – Specificity

ROC space

We plot the values of FPR along the horizontal axis (that is , x-axis) and the values of TPR along the vertical axis (that is, y-axis) in a plane. For each classifier, there is a unique point in this plane with coordinates (FPR,TPR). The ROC space is the part of the plane whose points correspond to (FPR,TPR). Each prediction result or instance of a confusion matrix represents one point in the ROC space.

The position of the point (FPR,TPR) in the ROC space gives an indication of the performance of the classifier. For example, let us consider some special points in the space.

Special points in ROC space

1. **The left bottom corner point (0, 0): Always negative prediction:**

   A classifier which produces this point in the ROC space never classifies an example as positive, neither rightly nor wrongly, because for this point TP = 0 and FP = 0. It always makes negative predictions. All positive instances are wrongly predicted and all negative instances are correctly predicted. It commits no false positive errors.

2. **The right top corner point (1, 1): Always positive prediction:**

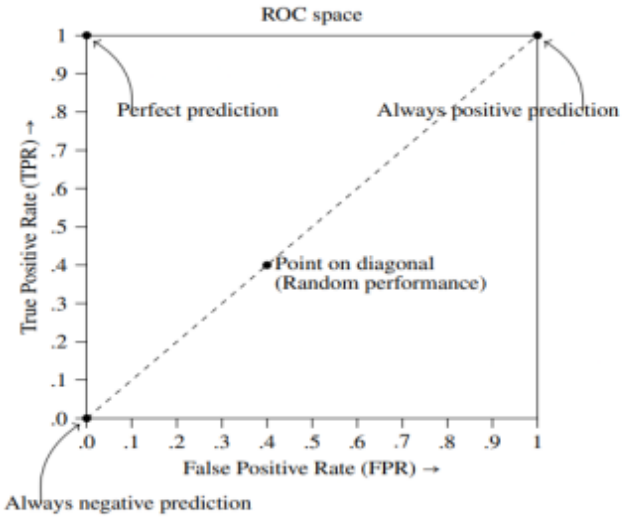
   A classifier which produces this point in the ROC space always classifies an example as positive because for this point FN = 0 and TN = 0. All positive instances are correctly predicted and all negative instances are wrongly predicted. It commits no false negative errors.

3. **The left top corner point (0, 1): Perfect prediction:**

   A classifier which produces this point in the ROC space may be thought as a perfect classifier. It produces no false positives and no false negatives.

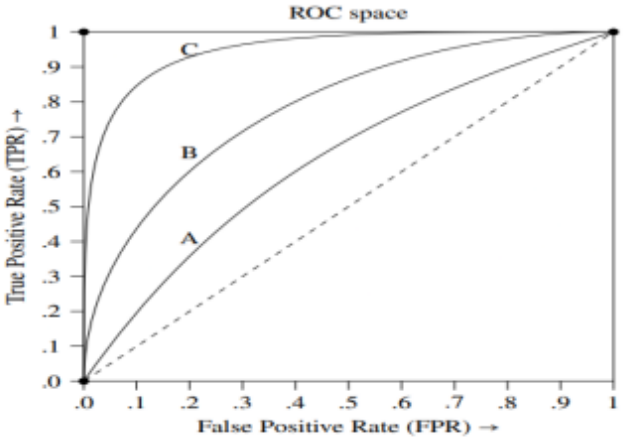
4. **Points along the diagonal: Random performance:**

   Consider a classifier where the class labels are randomly guessed, say by flipping a coin. Then, the corresponding points in the ROC space will be lying very near the diagonal line joining the points (0, 0) and (1, 1)



ROC curve

In the case of certain classification algorithms, the classifier may depend on a parameter. Different values of the parameter will give different classifiers and these in turn give different values to TPR and FPR. The ROC curve is the curve obtained by plotting in the ROC space the points (TPR , FPR) obtained by assigning all possible values to the parameter in the classifier



The closer the ROC curve is to the top left corner (0, 1) of the ROC space, the better the accuracy of the classifier. Among the three classifiers A, B, C with ROC curves as shown in Figure above, the classifier C is closest to the top left corner of the ROC space. Hence, among the three, it gives the best accuracy in predictions.

AUC: Area Under the ROC Curve

- AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).
- AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example.
- AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

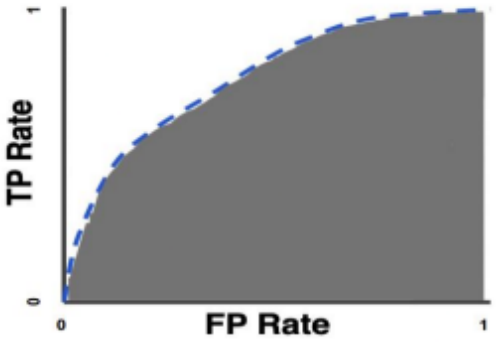


Figure 5. AUC (Area under the ROC Curve).