MAILAM Engineering College

Approved by AICTE, New Delhi, affiliated to Anna University, Chennai, Accredited by NBA & TCS

AL3451 – MACHINE LEARNING

[REGULATION-2021]

STUDY MATERIAL

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

NAME OF THE STUDENT:……

REGISTER NUMBER:……

YEAR / SEM:……

ACADEMIC YEAR:……

PREPARED BY

Mrs. G. Vasanthi, ASP/AI & DS

MAILAM Engineering College

Approved by AICTE, New Delhi, affiliated to Anna University, Chennai, Accredited by NBA & TCS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

AL3451 - MACHINE LEARNING

II Yr. / III SEM

SYLLABUS

COURSE OBJECTIVES:

- To understand the basic concepts of machine learning.
- To understand and build supervised learning models.
- To understand and build unsupervised learning models.
- To evaluate the algorithms based on corresponding metrics identified

UNIT I INTRODUCTION TO MACHINE LEARNING 8

Review of Linear Algebra for machine learning; Introduction and motivation for machine learning; Examples of machine learning applications, Vapnik-Chervonenkis (VC) dimension, Probably Approximately Correct (PAC) learning, Hypothesis spaces, Inductive bias, Generalization, Bias variance trade-off.

UNIT II SUPERVISED LEARNING 11

Linear Regression Models: Least squares, single & multiple variables, Bayesian linear regression, gradient descent, Linear Classification Models: Discriminant function – Perceptron algorithm, Probabilistic discriminative model - Logistic regression, Probabilistic generative model – Naive Bayes, Maximum margin classifier – Support vector machine, Decision Tree, Random Forests

UNIT III ENSEMBLE TECHNIQUES AND UNSUPERVISED LEARNING 9

Combining multiple learners: Model combination schemes, Voting, Ensemble Learning - bagging, boosting, stacking, Unsupervised learning: K-means, Instance Based Learning: KNN, Gaussian mixture models and Expectation maximization.

UNIT IV NEURAL NETWORKS 9

Multilayer perceptron, activation functions, network training – gradient descent optimization – stochastic gradient descent, error backpropagation, from shallow networks to deep networks –Unit saturation (aka the vanishing gradient problem) – ReLU, hyperparameter tuning, batch normalization, regularization, dropout.

UNIT V DESIGN AND ANALYSIS OF MACHINE LEARNING EXPERIMENTS 8

Guidelines for machine learning experiments, Cross Validation (CV) and resampling – K-fold CV, bootstrapping, measuring classifier performance, assessing a single classification algorithm and comparing two classification algorithms – t test, McNemar's test, K-fold CV paired t test.

COURSE OUTCOMES:

At the end of this course, the students will be able to:

CO1: Explain the basic concepts of machine learning.

CO2 : Construct supervised learning models.

CO3 : Construct unsupervised learning algorithms.

CO4: Evaluate and compare different models

TOTAL: 45 PERIODS

TEXTBOOKS:

1. Ethem Alpaydin, "Introduction to Machine Learning", MIT Press, Fourth Edition, 2020.

2. Stephen Marsland, "Machine Learning: An Algorithmic Perspective, "Second Edition", CRC Press, 2014.

REFERENCES:

1. Christopher M. Bishop, "Pattern Recognition and Machine Learning", Springer, 2006.

2. Tom Mitchell, "Machine Learning", McGraw Hill, 3rd Edition, 1997.

3. Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar, "Foundations of Machine Learning", Second Edition, MIT Press, 2012, 2018.

4. Ian Goodfellow, Yoshua Bengio, Aaron Courville, "Deep Learning", MIT Press, 2016

5. Sebastain Raschka, Vahid Mirjalili , "Python Machine Learning", Packt publishing, 3rd Edition, 2019.

ANNA UNIVERSITY UPDATED QP – AP 2024, ND 2024

Prepared by –

Mrs.G.Vasanthi, ASP/AI&DS

Verified By: Dr. S. Artheeswari, HOD/AI&DS

PRINCIPAL

MAILAM Engineering College

Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai,
Accredited by National Board of Accreditation (NBA), Accredited by NAAC with "A" Grade &
Accredited by TATA Consultancy Services (TCS), Chennai)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

II YEAR / IV SEM

AL3451- MACHINE LEARNING

UNIT I INTRODUCTION TO MACHINE LEARNING

SYLLABUS:

Review of Linear Algebra for machine learning; Introduction and motivation for machine learning; Examples of machine learning applications, Vapnik-Chervonenkis (VC) dimension, Probably Approximately Correct (PAC) learning, Hypothesis spaces, Inductive bias, Generalization, Bias variance trade-off.

PART A

1. Define linear Algebra.

- Linear Algebra is an essential field of mathematics, which defines the study of vectors, matrices, planes, mapping, and lines required for linear transformation.

2. What are the Benefits of learning Linear Algebra before Machine learning?

- Learning Linear Algebra before Machine Learning provides a strong foundation for understanding how machine learning algorithms work by enabling you to represent and manipulate data efficiently using vectors and matrices, which are core concepts in linear algebra, allowing for better intuition about data transformations, optimization techniques, and the inner workings of various models, especially in areas like dimensionality reduction and regression analysis.

3. Define Machine Learning. [Apr/May 2024]

- 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 ".
- Machine learning is programming computers to optimize a performance criterion using example data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data.

4. Mention the various classification of Machine Learning

Machine learning implementations are classified into four major categories, depending on the nature of the learning "signal" or "response" available to a learning system which are as follows:

- Supervised learning
- Unsupervised learning
- Reinforcement learning
- Semi-supervised learning

5. Define 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.
- The given data is labeled.
- Both classification and regression problems are supervised learning problems.

6. Define 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, classification or categorization is not included in the observations.
- In unsupervised learning the agent learns patterns in the input without any explicit feedback.
- The most common unsupervised learning task is clustering: detecting potentially useful clusters of input examples.

6. What is Reinforcement learning?

- In reinforcement learning the agent learns from a series of reinforcements: rewards and punishments.
- Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards.

- A learner 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.

7. What is Semi-supervised learning?

- Semi-Supervised learning is a type of Machine Learning algorithm that represents the intermediate ground between Supervised and Unsupervised learning algorithms.
- It uses the combination of labeled and unlabeled datasets during the training period, where an incomplete training signal is given: a training set with some of the target outputs missing.

8. How to categorize algorithm based on required Output?

- Classification
- Regression
- Clustering

9. How does linear algebra help in machine learning?

- The first step towards learning Math for ML is to learn linear algebra.
- Linear Algebra is the mathematical foundation that solves the problem of representing data as well as computations in machine learning models. It is the math of arrays technically referred to as vectors, matrices and tensors.

10. Is linear algebra done right good for machine learning?

- While 'Linear Algebra Done Right' is a remarkable primer for advanced theoretical courses, such as operator theory, functional analysis, or modern algebra, it might not suffice as a standalone resource for those looking to directly apply the concepts to real-world problems in data analysis and machine learning.

11. What is motivation in machine learning?

- In machine learning, motivation thrives on challenge and growth. Embrace tough problems as learning opportunities, seeking advice and experimenting with novel solutions.

12. Is ChatGPT, Alexa machine learning? Or Machine Learning Help Alexa and Siri Learn.

- ChatGPT is an extrapolation of a class of machine learning Natural Language Processing models known as Large Language Model (LLMs).

- Every time Alexa or Siri make a mistake when responding to your request, it uses the data it receives based on how it responded to the original query to improve the next time. If an error was made, it takes that data and learns from it.

13. What is the full form of GPT?

- GPT stands for Generative Pre-training Transformer. In essence, GPT is a kind of artificial intelligence (AI). When we talk about AI, we might think of sci-fi movies or robots. But AI is much more mundane and user-friendly.

14. What is the dimension of apnik Chervonenkis?

- In Vapnik–Chervonenkis theory, the Vapnik–Chervonenkis (VC) dimension is a measure of the size (capacity, complexity, expressive power, richness, or flexibility) of a class of sets.

15. What is the VC dimension of SVM?

- "The VC Dimension of affine classifiers of the form f(x)=w·x+b in n dimensions − i.e. w∈Rn − is n+1": this corresponds to the case of what is called a linear SVM. "The VC Dimension of an SVM equipped with an RBF kernel is infinite."

16. What is hypothesis space search in genetic algorithm?

- Hypothesis space search is a key concept in machine learning that refers to the process of searching for the best possible model or hypothesis that can fit the given data.
- In machine learning, the hypothesis space is the set of all possible models that the algorithm can learn from the data.

17. What is the purpose of restricting hypothesis space in machine learning?

- The purpose of restricting hypothesis space in machine learning is so that these can fit well with the general data that is needed by the user. It checks the reality or deception of observations or inputs and examinations them appropriately.

18. What is the hypothesis space in neural network?

- The hypothesis space comprises all possible legal hypotheses that a machine learning algorithm can consider. Hypotheses are formulated based on various algorithms and techniques, including linear regression, decision trees, and neural networks.

19. What is bias in ML?

- Bias in ML is an sort of mistake in which some aspects of a dataset are given more weight and/or representation than others. A skewed outcome, low accuracy levels, and analytical errors result from a dataset that is biased that does not represent a model's use case accurately.

20. What is inductive bias in machine learning?

- Inductive bias is anything which makes the algorithm learn one pattern instead of another pattern (e.g., step-functions in decision trees instead of continuous functions in linear regression models).
- Learning involves searching a space of solutions for a solution that provides a good explanation of the data.

21. Write some examples for Machine Learning Applications. [Apr/May 2024]

- Facial recognition.
- Product recommendations.
- Email automation and spam filtering.
- Financial accuracy.
- Social media optimization.
- Healthcare advancement.
- Mobile voice to text and predictive text.

22. Define Generalization in Machine Learning. [Nov/Dec 2024]

- Generalization in machine learning is a model's ability to perform well on new data that it hasn't seen before. It's a key technique that allows machine learning models to make accurate predictions on unseen data.

23. What is Bias in Machine Learning? [Nov/Dec 2024]

- Bias in machine learning threatens the accuracy, fairness, and reliability of AI systems, leading to flawed outcomes that impact industries and individuals alike.
- Understanding the nature of bias, its origins, and its effects is critical for designing machine learning models that drive equitable and effective results.

PART-B

1. Explain in detail about Review of Linear Algebra for machine learning.

- Linear algebra is essential for many machine learning algorithms and techniques. It helps in manipulating and processing data, which is often represented as vectors and matrices. These mathematical tools make computations faster and reveal patterns within the data.
- It simplifies complex tasks like data transformation, dimensionality reduction (e.g., PCA), and optimization. Key concepts like matrix multiplication, eigenvalues, and linear transformations help in training models and improving predictions efficiently.

Fundamental Concepts in Linear Algebra for Machine Learning

- In machine learning, vectors, matrices, and scalars play key roles in handling and processing data.
- Vectors are used to represent individual data points, where each number in the vector corresponds to a specific features of the dataset (like age, income, or hours).
- Matrices are considered as data storage units used to store large datasets, with rows representing different data points and columns representing features.
- Scalars are single numbers that scale vectors or matrices, often used in algorithms like gradient descent to adjust the weights or learning rate, helping the model improve over time.
- Together, these mathematical tools enable efficient computation, pattern recognition, and model training in machine learning.

1. Vectors

Vectors are quantities that have both magnitude and direction, often represented as arrows in space.

$$\mathbf{v} = \begin{bmatrix} 2 \\ -1 \\ 4 \end{bmatrix}$$

2. Matrices

Matrices are rectangular arrays of numbers, arranged in rows and columns.

Matrices are used to represent linear transformations, systems of linear equations, and data transformations in machine learning.

Example: $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ u = [3, 4] v = [-1, 2].

3. Scalars

Scalars are single numerical values, without direction, magnitude only. Scalars are just single numbers that can multiply vectors or matrices. In machine learning, they're used to adjust things like the weights in a model or the learning rate during training

- Example: Let's consider a scalar, **k= 3,** and a vector $[\mathbf{v} = \begin{bmatrix} 2 \\ -1 \\ 4 \end{bmatrix}]$

Scalar multiplication involves multiplying each component of the vector by the scalar. So, if we multiply the vector v by the scalar k=3 we get:

$$k \cdot \mathbf{v} = 3 \cdot \begin{bmatrix} 2 \\ -1 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \cdot 2 \\ 3 \cdot (-1) \\ 3 \cdot 4 \end{bmatrix} = \begin{bmatrix} 6 \\ -3 \\ 12 \end{bmatrix}$$

Operations in Linear Algebra

1. Addition and Subtraction

Addition and subtraction of vectors or matrices involve adding or subtracting corresponding elements.

Example: let's consider we have got 2 vectors, u and v :

$$[\mathbf{u} = \begin{bmatrix} 2 \\ -1 \\ 4 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 3 \\ 0 \\ -2 \end{bmatrix}]$$

addition: $[\mathbf{u} + \mathbf{v} = \begin{bmatrix} 2 \\ -1 \\ 4 \end{bmatrix} + \begin{bmatrix} 3 \\ 0 \\ -2 \end{bmatrix} = \begin{bmatrix} 2+3 \\ -1+0 \\ 4+(-2) \end{bmatrix} = \begin{bmatrix} 5 \\ -1 \\ 2 \end{bmatrix}]$

subtraction: $[\mathbf{u}-\mathbf{v} = \begin{bmatrix} 2 \\ -1 \\ 4 \end{bmatrix} - \begin{bmatrix} 3 \\ 0 \\ -2 \end{bmatrix} = \begin{bmatrix} 2-3 \\ -1-0 \\ 4-(-2) \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ 6 \end{bmatrix}]$

2. Scalar Multiplication

Scalar multiplication involves multiplying each element of a vector or matrix by a scalar.

Example: Consider the scalar k=3 and a vector

$$\mathbf{v} = \begin{bmatrix} 2 \\ -1 \\ 4 \end{bmatrix}$$

Scalar multiplication involves multiplying each component of the vector by the scalar. So, if we multiply the vector v by the scalar k=3 , we get :

$$k \cdot \mathbf{v} = 3 \cdot \begin{bmatrix} 2 \\ -1 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \cdot 2 \\ 3 \cdot (-1) \\ 3 \cdot 4 \end{bmatrix} = \begin{bmatrix} 6 \\ -3 \\ 12 \end{bmatrix}$$

3. Dot Product (Scalar Product)

The dot product of two vectors tells us how similar their directions are. To calculate it, you multiply the matching elements of the vectors and then add them together.

Example: For example, given two vectors($\mathbf{u} - [u_1, u_2, u_3]$ and $\mathbf{v} - [v_1, v_2, v_3]$), their dot product is calculated as:

$$\mathbf{u} \cdot \mathbf{v} - u_1 \cdot v_1 + u_2 \cdot v_2 + u_3 \cdot v_3$$

4. Cross Product (Vector product)

The cross product of two 3D vectors makes a new vector that points at a right angle to the two original vectors. It is used less frequently in machine learning compared to the dot product.

Example: Given two vectors u and v, their cross product u×v is calculated as:

$$\mathbf{u} \times \mathbf{v} - \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \times \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} - \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix}$$

Linear Transformations

- Linear transformations are basic operations in linear algebra that change vectors and matrices while keeping important properties like straight lines and proportionality.
- In machine learning, they are key for tasks like preparing data, creating features, and training models. This section covers the definition, types, and uses of linear transformations.

A. Definition and Explanation

- Linear transformations are functions that map vectors from one vector space to another in a linear manner.
- Formally, a transformation T is considered linear if it satisfies two properties:
- Additivity: T(u+v)=T(u)+T(v) for all vectors u and v.
- Homogeneity: T(kv)=kT(v) for all vectors v and scalars k.

B. Common Linear Transformations in Machine Learning

- Common linear transformations in machine learning are operations that help manipulate data in useful ways, making it easier for models to learn patterns and make predictions. Some common linear transformations are:
- **Translation:** Translation means moving data points around without changing their shape or size. In machine learning, this is often used to center data by subtracting the average value from each data point.
- **Scaling:** Scaling involves stretching or compressing vectors along each dimension. It is used in feature scaling to make sure all features are on a similar scale, so one feature doesn't dominate the model.
- **Rotation:** Rotation involves turning data around a point or axis. It's not used much in basic machine learning but can be helpful in fields like computer vision and robotics.

Matrix Operations

- Matrix operations are a key part of linear algebra and are vital for handling and analyzing data in machine learning. This section covers important operations like multiplication, transpose, inverse, and determinant, and explains their importance and how they are used.

Lets dive into some common matrix operations.

- Matrix multiplication is a fundamental operation in linear algebra, involving the multiplication of two matrices to produce a new matrix. Given two matrices A and B, the product matrix C=A·B is computed by taking the dot product of each row of matrix A with each column of matrix B.
- Matrix multiplication is widely used in machine learning for various tasks, including transformation of feature vectors, computation of model parameters, and neural network operations such as feedforward and backpropagation.

Example:

Lets consider we have two matrices A and B,

$$[A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 3 & 0 \\ 1 & 2 \end{bmatrix}]$$

- To multiply matrices (A) and (B) we have to perform row-column multiplication.
- The element at the row (i) and the column (j) of the resulting matrix (c) is calculated by taking the dot product of the (i)th row of matrix (A) and the (j)th column of matrix (B).

For example, the element c11 of matrix (c) is calculated as:

$$[c_{12} = a_{11} \cdot b_{12} + a_{12} \cdot b_{22} = 2 \cdot 0 + 1 \cdot 2 = 2]$$

Following this pattern, we can calculate all elements of matrix (C) is

$$[C = \begin{bmatrix} 7 & 2 \\ 5 & 4 \end{bmatrix}]$$

So, the result of the matrix multiplication

$(A \times B)$ is (c).

B. Transpose and Inverse of Matrices

Transpose:

- The transpose of a matrix involves flipping its rows and columns, resulting in a new matrix where the rows become columns and vice versa.
- It is denoted by AT, and its dimensions are the reverse of the original matrix.

Inverse:

- The inverse of a square matrix A is another matrix denoted by A−1 such that A·A−1=I ,where I is the identity matrix.
- Not all matrices have inverses, and square matrices with a determinant not equal to zero are invertible.
- Inverse matrices are used in solving systems of linear equations, computing solutions to optimization problems, and performing transformations.

C. Determinants

- A determinant is a number that comes from a square matrix. It helps tell us if the matrix can be flipped or not. If the determinant is zero, the matrix can't be flipped. If it's not zero, it means the matrix can be inverted or reversed.
- Significance: The determinant of a matrix tells us if it can be inverted (flipped) and how it transforms space. If the determinant is zero, the matrix can't be inverted.
- Properties: The determinant satisfies several properties, including linearity, multiplicativity, and the property that a matrix is invertible if and only if its determinant is non-zero.

Eigenvalues and Eigenvectors

- Eigenvalues and eigenvectors are fundamental concepts in linear algebra that play a significant role in machine learning algorithms and applications. In this section, we explore the definition, significance, and applications of eigenvalues and eigenvectors.

- Eigenvalues of a square matrix A are scalar values that represent how a transformation represented by A stretches or compresses vectors in certain directions.
- Eigenvalues quantify the scale of transformation along the corresponding eigenvectors and are crucial for understanding the behavior of linear transformations.

Example: Consider the matrix:

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

To find the eigenvalues , λ , we solve the characteristic equation:

$$\det(A - \lambda I) = 0$$

Substituting the values:

$$\det\left(\begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix}\right) = 0$$

This simplifies to:

$$(2-\lambda)^2 - 1 = 0$$

Solving this, we find $(\lambda_1 = 1)$ and $(\lambda_2 = 3)$

for $(\lambda_1 = 1)$, solving $((A - \lambda_1 I)\mathbf{v}_1 = \mathbf{0})$, we find the eigenvector $(\mathbf{v}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix})$

for $(\lambda_2 = 3)$, solving $((A - \lambda_2 I)\mathbf{v}_2 = \mathbf{0})$, we find the eigenvector $(\mathbf{v}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix})$

- Eigenvectors are non-zero vectors that are transformed by a matrix only by a scalar factor, known as the eigenvalue. They represent the directions in which a linear transformation represented by a matrix stretches or compresses space.
- Eigenvectors corresponding to distinct eigenvalues are linearly independent and form a basis for the vector space.

Eigen Decomposition

- Eigen decomposition is the process of decomposing a square matrix into its eigenvalues and eigenvectors.
- It is expressed as

$$A = Q\Lambda Q^{-1},$$

- where Q is a matrix whose columns are the eigenvectors of A, and λ is a diagonal matrix containing the corresponding eigenvalues.

- Eigen decomposition provides insights into the structure and behavior of linear transformations, facilitating various matrix operations and applications in machine learning.

Applications in Machine Learning

- **Dimensionality Reduction:** Techniques like Principal Component Analysis (PCA) use eigenvalues and eigenvectors to find the most important directions in high-dimensional data and reduce it to fewer dimensions. The eigenvalues tell us how much variance (or information) each direction explains, helping us keep the important parts while simplifying the data.
- **Matrix Factorization:** Methods like Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF) use eigenvalue decomposition to break down large matrices into smaller, more manageable parts. This helps us extract important features from complex data, making analysis more efficient.

2. Explain in detail about Applications of Linear Algebra in Machine Learning.

Applications of Linear Algebra in Machine Learning

- Linear algebra serves as the backbone of many machine learning algorithms, providing powerful tools for data manipulation, model representation, and optimization.
- In this section, we explore some of the key applications of linear algebra in machine learning, including principal component analysis (PCA), singular value decomposition (SVD), linear regression, support vector machines (SVM), and neural networks.

A. Principal Component Analysis (PCA)

- Principal Component Analysis (PCA) is a dimensionality reduction technique that utilizes linear algebra to identify the principal components in high-dimensional data. The main steps of PCA involve:
- **Covariance Matrix Calculation:** Compute the covariance matrix of the data to understand the relationships between different features.
- **Eigenvalue Decomposition:** Decompose the covariance matrix into its eigenvalues and eigenvectors to identify the principal components.

- **Projection onto Principal Components:** Project the original data onto the principal components to reduce the dimensionality while preserving the maximum variance.

B. Singular Value Decomposition (SVD)

- Singular Value Decomposition (SVD) is a matrix factorization technique widely used in machine learning for dimensionality reduction, data compression, and noise reduction. The key steps of SVD include:
- **Decomposition:** Decompose the original matrix into the product of three matrices:
- $A=U\Sigma V^T$ where U and V are orthogonal matrices, and σ is a diagonal matrix of singular values.
- **Dimensionality Reduction:** Retain only the most significant singular values and their corresponding columns of U and V to reduce the dimensionality of the data.

C. Linear Regression

- Linear regression is a supervised learning algorithm used for modeling the relationship between a dependent variable and one or more independent variables. Linear algebra plays a crucial role in solving the linear regression problem efficiently through techniques such as:

  Matrix Formulation: Representing the linear regression problem in matrix form $Y=X\beta+\epsilon$ where Y is the dependent variable, X is the matrix of independent variables, β is the vector of coefficients, and ϵ\epsilonϵ is the error term.

  Normal Equation: Solving the normal equation $X^TX\beta=X^T Y$ using linear algebra to obtain the optimal coefficients β.

D. Support Vector Machines (SVM)

- Support Vector Machines (SVM) are powerful supervised learning models used for classification and regression tasks. Linear algebra plays a crucial role in SVMs through:
- **Kernel Trick:** The kernel trick uses linear algebra to map data into higher dimensions, allowing SVMs to handle complex, non-linear problems like classification. Optimization: In SVM, optimization involves finding the best decision boundary. This is done by turning the problem into a math problem and solving it using linear algebra methods, making the process faster and more efficient.
- **E. Neural Networks**

- Neural networks, especially deep learning models, heavily rely on linear algebra for model representation, parameter optimization, and forward/backward propagation. Key linear algebraic operations in neural networks include:
- **Matrix Multiplication:** Performing matrix multiplication operations between input features and weight matrices in different layers of the neural network during the forward pass.
- **Gradient Descent:** Computing gradients efficiently using backpropagation and updating network parameters using gradient descent optimization algorithms, which involve various linear algebraic operations.
- **Weight Initialization:** Initializing network weights using techniques such as Xavier initialization and He initialization, which rely on linear algebraic properties for proper scaling of weight matrices.

Conclusion

- Linear algebra is fundamental to machine learning, offering essential tools for data manipulation and algorithm development.
- Concepts like vectors, matrices, and techniques such as eigenvalue decomposition and singular value decomposition are key to algorithms used in dimensionality reduction, regression, classification, and neural networks.
- Mastering linear algebra is crucial for success in machine learning and AI, and its importance will keep growing as the field advances.

2. Explain in detail about Introduction and Motivation of Machine Learning.

- Machine learning (ML) allows computers to learn and make decisions without being explicitly programmed.
- It involves feeding data into algorithms to identify patterns and make predictions on new data.
- Machine learning is used in various applications, including image and speech recognition, natural language processing, and recommender systems in fig.1.1.

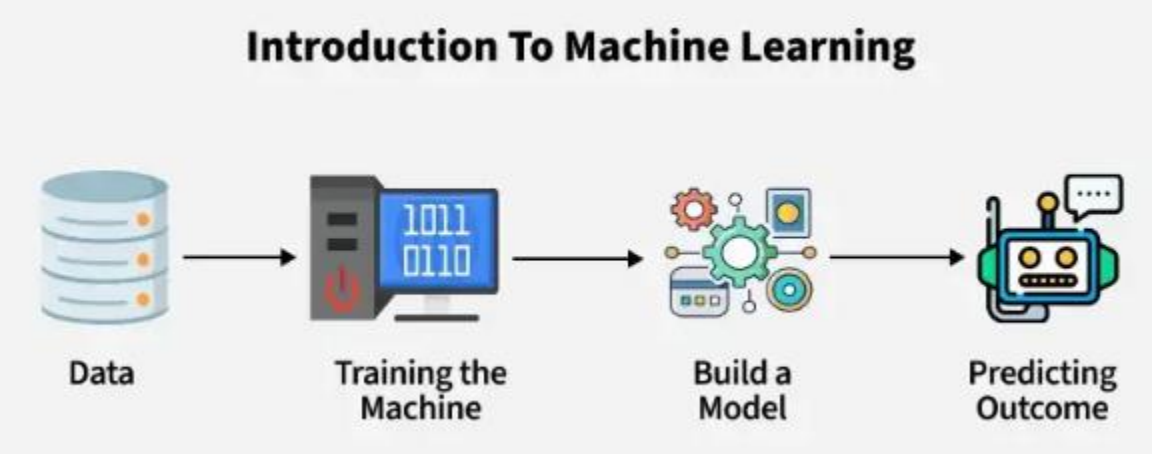


Fig.1.1 Introduction to Machine Learning

Why do we need Machine Learning?

- **Machine Learning** algorithm learns from data, train on patterns, and solve or predict complex problems beyond the scope of traditional programming. It drives better decision-making and tackles intricate challenges efficiently.

Here's why ML is indispensable across industries:

1. Solving Complex Business Problems

- Traditional programming struggles with tasks like image recognition, natural language processing (NLP), and medical diagnosis. ML, however, thrives by learning from examples and making predictions without relying on predefined rules.

  Example Applications:

  - Image and speech recognition in healthcare.
  - Language translation and sentiment analysis.

2. Handling Large Volumes of Data

- With the internet's growth, the data generated daily is immense. ML effectively processes and analyzes this data, extracting valuable insights and enabling real-time predictions.

  Use Cases:

- Fraud detection in financial transactions.
- Social media platforms like Facebook and Instagram predicting personalized feed recommendations from billions of interactions.

3. Automate Repetitive Tasks

- ML automates time-intensive and repetitive tasks with precision, reducing manual effort and error-prone systems.

Examples:

- **Email Filtering**: Gmail uses ML to keep your inbox spam-free.
- **Chatbots**: ML-powered chatbots resolve common issues like order tracking and password resets.
- **Data Processing**: Automating large-scale invoice analysis for key insights.

4. Personalized User Experience

- ML enhances user experience by tailoring recommendations to individual preferences. Its algorithms analyze user behavior to deliver highly relevant content.

Real-World Applications:

- **Netflix**: Suggests movies and TV shows based on viewing history.
- **E-Commerce**: Recommends products you're likely to purchase.

5. Self Improvement in Performance

- ML models evolve and improve with more data, making them smarter over time. They adapt to user behavior and refine their performance.

Examples:

- **Voice Assistants** (e.g., Siri, Alexa): Learn user preferences, improve voice recognition, and handle diverse accents.
- **Search Engines**: Refine ranking algorithms based on user interactions.
- **Self-Driving Cars**: Enhance decision-making using millions of miles of data from simulations and real-world driving.

What Makes a Machine "Learn"?

- A machine "learns" by recognizing patterns and improving its performance on a task based on data, without being explicitly programmed.

The process involves:

1. **Data Input:** Machines require data (e.g., text, images, numbers) to analyze.
2. **Algorithms:** Algorithms process the data, finding patterns or relationships.
3. **Model Training:** Machines learn by adjusting their parameters based on the input data using mathematical models.
4. **Feedback Loop:** The machine compares predictions to actual outcomes and corrects errors (via optimization methods like gradient descent).
5. **Experience and Iteration:** Repeating this process with more data improves the machine's accuracy over time.
6. **Evaluation and Generalization:** The model is tested on unseen data to ensure it performs well on real-world tasks.

In essence, machines "learn" by continuously refining their understanding through data-driven iterations, much like humans learn from experience.

Importance of Data in Machine Learning

Data is the foundation of machine learning (ML). Without quality data, ML models cannot learn, perform, or make accurate predictions.

- Data provides the examples from which models learn patterns and relationships.
- High-quality and diverse data improves model accuracy and generalization.
- Data ensures models understand real-world scenarios and adapt to practical applications.
- Features derived from data are critical for training models.
- Separate datasets for validation and testing assess how well the model performs on unseen data.
- Data fuels iterative improvements in ML models through feedback loops.

Types of Machine Learning

1. Supervised learning

- Supervised learning is a type of machine learning where a model is trained on labeled data—meaning each input is paired with the correct output.
- The model learns by comparing its predictions with the actual answers provided in the training data.
- **Both *classification* and *regression* problems are supervised learning problems.**
- **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	sick
F	32	healthy
M	34	healthy
M	21	healthy

- In this example, supervised learning is to use this labeled data to train a model that can predict the label ("healthy" or "sick") for new patients based on their gender and age.

- For instance, if a new patient (e.g., Male, 50 years old) visits the clinic, the model can classify whether the patient is "healthy" or "sick" based on the patterns it learned during training.

2. Unsupervised learning:

- Unsupervised learning algorithms draw inferences from datasets consisting of input data without labeled responses. In unsupervised learning algorithms, classification or categorization is not included in the observations.
- **Example:** Consider the following data regarding patients entering a clinic. The dataset includes **unlabeled data**, where only the gender and age of the patients are available, with no health status labels.

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

- Here, unsupervised learning technique will be used to find patterns or groupings in the data such as clustering patients by age or gender.
- For example, the algorithm might group patients into clusters, such as "younger healthy patients" or "older patients," without prior knowledge of their health status.

3. Reinforcement Learning

- Reinforcement Learning (RL) trains an agent to act in an environment by maximizing rewards through trial and error. Unlike other machine learning types, RL doesn't provide explicit instructions.

  Instead, the agent learns by:

  - **Exploring Actions**: Trying different actions.
  - **Receiving Feedback**: Rewards for correct actions, punishments for incorrect ones.
  - **Improving Performance**: Refining strategies over time.

Example: Identifying a Fruit in fig.1.2

- The system receives an input (e.g., an apple) and initially makes an incorrect prediction ("It's a mango"). Feedback is provided to correct the error ("Wrong! It's an apple"), and the system updates its model based on this feedback.
- Over time, it learns to respond correctly ("It's an apple") when encountering similar inputs, improving accuracy through trial, error, and feedback.

Fig.1.2 Example Reinforcement Learning

Benefits of Machine Learning

- Enhanced Efficiency and Automation: ML automates repetitive tasks, freeing up human resources for more complex work. It also streamlines processes, leading to increased efficiency and productivity.
- Data-Driven Insights: ML can analyze vast amounts of data to identify patterns and trends that humans might miss. This allows for better decision-making based on real-world data.
- Improved Personalization: ML personalizes user experiences across various platforms. From recommendation systems to targeted advertising, ML tailors content and services to individual preferences.
- Advanced Automation and Robotics: ML empowers robots and machines to perform complex tasks with greater accuracy and adaptability. This is revolutionizing fields like manufacturing and logistics.

Challenges of Machine Learning

- Data Bias and Fairness: ML algorithms are only as good as the data they are trained on. Biased data can lead to discriminatory outcomes, requiring careful data selection and monitoring of algorithms.
- Security and Privacy Concerns: As ML relies heavily on data, security breaches can expose sensitive information. Additionally, the use of personal data raises privacy concerns that need to be addressed.
- Interpretability and Explainability: Complex ML models can be difficult to understand, making it challenging to explain their decision-making processes. This lack of transparency can raise questions about accountability and trust.
- Job Displacement and Automation: Automation through ML can lead to job displacement in certain sectors. Addressing the need for retraining and reskilling the workforce is crucial.

3. Explain in detail about Examples of machine learning applications.

- Machine learning is a buzzword for today's technology, and it is growing very rapidly day by day. We are using machine learning in our daily life even without knowing it

such as Google Maps, Google assistant, Alexa, etc. Below are some most trending real-world applications of Machine Learning in fig.1.3:

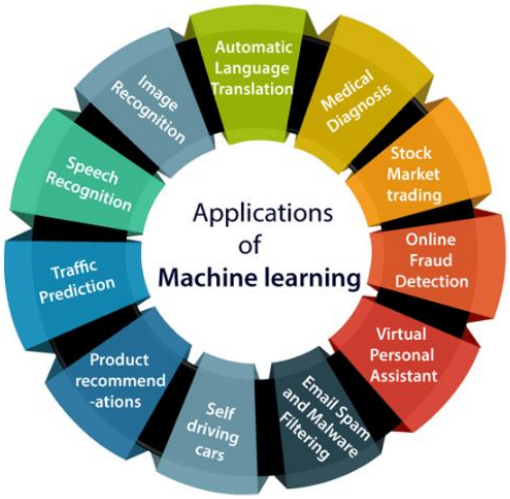


Fig.1.3 Applications of Machine Learning

1. Image Recognition:

- Image recognition is one of the most common applications of machine learning. It is used to identify objects, persons, places, digital images, etc. The popular use case of image recognition and face detection is, Automatic friend tagging suggestion:
- Facebook provides us a feature of auto friend tagging suggestion. Whenever we upload a photo with our Facebook friends, then we automatically get a tagging suggestion with name, and the technology behind this is machine learning's face detection and recognition algorithm.
- It is based on the Facebook project named "Deep Face," which is responsible for face recognition and person identification in the picture.

2. Speech Recognition

- While using Google, we get an option of "Search by voice," it comes under speech recognition, and it's a popular application of machine learning.
- Speech recognition is a process of converting voice instructions into text, and it is also known as "Speech to text", or "Computer speech recognition." At present, machine learning algorithms are widely used by various applications of speech recognition. Google assistant, Siri, Cortana, and Alexa are using speech recognition technology to follow the voice instructions.

3. Traffic prediction:

- If we want to visit a new place, we take help of Google Maps, which shows us the correct path with the shortest route and predicts the traffic conditions.
- It predicts the traffic conditions such as whether traffic is cleared, slow-moving, or heavily congested with the help of two ways:
- Real Time location of the vehicle form Google Map app and sensors
- Average time has taken on past days at the same time.
- Everyone who is using Google Map is helping this app to make it better. It takes information from the user and sends back to its database to improve the performance.

4. Product recommendations:

- Machine learning is widely used by various e-commerce and entertainment companies such as Amazon, Netflix, etc., for product recommendation to the user. Whenever we search for some product on Amazon, then we started getting an advertisement for the same product while internet surfing on the same browser and this is because of machine learning.
- Google understands the user interest using various machine learning algorithms and suggests the product as per customer interest.
- As similar, when we use Netflix, we find some recommendations for entertainment series, movies, etc., and this is also done with the help of machine learning.

5. Self-driving cars:

- One of the most exciting applications of machine learning is self-driving cars. Machine learning plays a significant role in self-driving cars. Tesla, the most popular car manufacturing company is working on self-driving car. It is using unsupervised learning method to train the car models to detect people and objects while driving.

6. Email Spam and Malware Filtering:

- Whenever we receive a new email, it is filtered automatically as important, normal, and spam.
- We always receive an important mail in our inbox with the important symbol and spam emails in our spam box, and the technology behind this is Machine learning. Below are some spam filters used by Gmail:

- Content Filter
- Header filter
- General blacklists filter
- Rules-based filters
- Permission filters
- Some machine learning algorithms such as Multi-Layer Perceptron, Decision tree, and Naïve Bayes classifier are used for email spam filtering and malware detection.

7. Virtual Personal Assistant:

- We have various virtual personal assistants such as Google assistant, Alexa, Cortana, Siri. As the name suggests, they help us in finding the information using our voice instruction. These assistants can help us in various ways just by our voice instructions such as Play music, call someone, Open an email, Scheduling an appointment, etc.
- These virtual assistants use machine learning algorithms as an important part.
- These assistant record our voice instructions, send it over the server on a cloud, and decode it using ML algorithms and act accordingly.

8. Online Fraud Detection:

- Machine learning is making our online transaction safe and secure by detecting fraud transaction. Whenever we perform some online transaction, there may be various ways that a fraudulent transaction can take place such as fake accounts, fake ids, and steal money in the middle of a transaction. So to detect this, Feed Forward Neural network helps us by checking whether it is a genuine transaction or a fraud transaction.
- For each genuine transaction, the output is converted into some hash values, and these values become the input for the next round. For each genuine transaction, there is a specific pattern which gets change for the fraud transaction hence, it detects it and makes our online transactions more secure.

9. Stock Market trading:

- Machine learning is widely used in stock market trading. In the stock market, there is always a risk of up and downs in shares, so for this machine learning's long short term memory neural network is used for the prediction of stock market trends.

10. Medical Diagnosis:

- In medical science, machine learning is used for diseases diagnoses. With this, medical technology is growing very fast and able to build 3D models that can predict the exact position of lesions in the brain.
- It helps in finding brain tumors and other brain-related diseases easily.

11. Automatic Language Translation:

- Nowadays, if we visit a new place and we are not aware of the language then it is not a problem at all, as for this also machine learning helps us by converting the text into our known languages. Google's GNMT (Google Neural Machine Translation) provide this feature, which is a Neural Machine Learning that translates the text into our familiar language, and it called as automatic translation.
- The technology behind the automatic translation is a sequence to sequence learning algorithm, which is used with image recognition and translates the text from one language to another language.

4. Explain in detail about Vapnik-Chervonenkis (VC) dimension.

Illustrate the Vapnik-Chervonenkis (VC) dimension. [Nov/Dec 2024]

Discuss the following [Apr/May 2024]

- **Vapnik-Chervonenkis Dimension**
- **Probably Approximately Correct Learning**

- The Vapnik-Chervonenkis (VC) dimension is a measure of the capacity of a hypothesis set to fit different data sets. It was introduced by Vladimir Vapnik and Alexey Chervonenkis in the 1970s and has become a fundamental concept in statistical learning theory. The VC dimension is a measure of the complexity of a model, which can help us understand how well it can fit different data sets.
- The VC dimension of a hypothesis set H is the largest number of points that can be shattered by H. A hypothesis set H shatters a set of points S if, for every possible labeling of the points in S, there exists a hypothesis in H that correctly classifies the points. In other words, a hypothesis set shatters a set of points if it can fit any possible labeling of those points.

Bounds of VC – Dimension

- The VC dimension provides both upper and lower bounds on the number of training examples required to achieve a given level of accuracy. The upper bound on the number of training examples is logarithmic in the VC dimension, while the lower bound is linear.
- VC dimension, short for Vapnik-Chervonenkis dimension, is a measure of the complexity of a machine learning model. It is named after the mathematicians Vladimir Vapnik and Alexey Chervonenkis, who developed the concept in the 1970s as part of their work on statistical learning theory.
- VC dimension is defined as the largest number of points that can be shattered by a binary classifier without misclassification. In other words, it is a measure of the model's capacity to fit arbitrary labeled datasets. The more complex the model, the higher its VC dimension.

Mathematically, the VC dimension of a binary classifier is defined as follows:

- Given a set of n points S = {x1, x2, ..., xn} in a d-dimensional space and a binary classifier h, the VC dimension of h is the largest integer d such that there exists a set of d points that can be shattered by h, i.e., for any labeling of the d points, there exists a hypothesis h in H that correctly classifies them.
- Formally, the VC dimension of h is:
- VC(h) = max{d | there exists a set of d points that can be shattered by h}
- VC dimension has important implications for machine learning models. It is related to the model's generalization ability, i.e., its ability to perform well on unseen data. A model with a low VC dimension is less complex and is more likely to generalize well, while a model with a high VC dimension is more complex and is more likely to overfit the training data in fig.1.4.

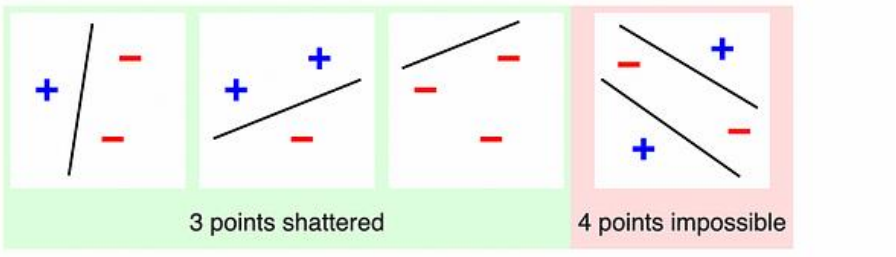


Fig.1.4 VC Dimension of a Binary Classifier

- VC dimension is used in various areas of machine learning, such as support vector machines (SVMs), neural networks, decision trees, and boosting algorithms. In SVMs, the VC dimension is used to bound the generalization error of the model. In neural networks, the VC dimension is related to the number of parameters in the model and is used to determine the optimal number of hidden layers and neurons. In decision trees, the VC dimension is used to measure the complexity of the tree and to prevent overfitting.

Limitations to VC Dimension:

- However, there are some limitations to VC dimension. First, it only applies to binary classifiers and cannot be used for multi-class classification or regression problems.
- Second, it assumes that the data is linearly separable, which is not always the case in real-world datasets. Third, it does not take into account the distribution of the data and the noise level in the dataset.

Most Powerful Model:

- The most powerful model according to VC dimension is the SVM with a Gaussian kernel, which has an infinite VC dimension. This means that the model has the capacity to fit any labeled dataset perfectly.
- However, this does not necessarily mean that it is the best model for all problems, as it can be computationally expensive and may not generalize well to unseen data.

Finite VC Dimension:

- On the other hand, decision trees have a finite VC dimension, which makes them less complex and more likely to generalize well. However, they may not be suitable for datasets with high dimensionality or complex decision boundaries.
- In summary, VC dimension is a powerful tool for measuring the complexity of machine learning models and for understanding their generalization ability. It has important implications for model selection, regularization, and optimization. However, it has some limitations and should be used in conjunction with other evaluation metrics and techniques.

Applications of VC – Dimension

- The VC dimension has a wide range of applications in machine learning and statistics.

- **For example**, it is used to analyze the complexity of neural networks, support vector machines, and decision trees. The VC dimension can also be used to design new learning algorithms that are robust to noise and can generalize well to unseen data.
- The VC dimension can be extended to more complex learning scenarios, such as multiclass classification and regression. The concept of the VC dimension can also be applied to other areas of computer science, such as computational geometry and graph theory.

5. Explain in detail about Probably Approximately Correct (PAC) learning.

Discuss the following **[Apr/May 2024]**

- **Vapnik-Chervonenkis Dimension**
- **Probably Approximately Correct Learning**

- In the rapidly evolving field of machine learning, understanding the theoretical underpinnings is crucial for developing robust algorithms.
- One of the foundational concepts in this domain is **Probably Approximately Correct (PAC) learning**. Introduced by Leslie Valiant in 1984, PAC learning provides a framework for analyzing how learning algorithms can generalize from a finite set of training examples to unseen instances.
- This blog post aims to explore the intricacies of PAC learning in machine learning, its significance, core concepts, and practical applications, all while embedding relevant coding terms and code snippets.

What is PAC Learning?

- PAC learning is a theoretical framework that addresses the question of how much data is necessary for a learning algorithm to perform well on new, unseen data.
- The core idea is that a learning algorithm can be considered PAC if, given a sufficient number of training samples, it can produce a hypothesis that is likely (with high probability) to be approximately correct (within a specified error margin).

Key Components of PAC Learning

1. **Hypothesis Space**: This is the set of all possible hypotheses that a learning algorithm can choose from. The complexity of the hypothesis space significantly impacts the sample complexity required for learning.
2. **Sample Complexity**: This refers to the number of training examples needed to ensure that the learned hypothesis will generalize well to new data. In PAC learning, it is crucial to determine how many samples are required to achieve a desired level of accuracy and confidence.
3. **Generalization**: This is the ability of a learning algorithm to perform well on unseen data. In the PAC framework, generalization is quantified by the probability that the chosen hypothesis will have an error rate within an acceptable range on new samples.
4. **Error Rate**: The error rate is defined as the probability that the hypothesis will misclassify an example drawn from the underlying distribution. PAC learning aims to minimize this error rate while ensuring that the hypothesis is consistent with the training data.

The PAC Learning Theorem

- The **PAC learning theorem** provides formal guarantees about the performance of learning algorithms.
- It states that for a given accuracy (ε) and confidence (δ), there exists a sample size (m) such that any learning algorithm that returns a hypothesis consistent with the training samples will, with probability at least 1−δ1−δ, have an error rate less than ε on unseen data. Mathematically, this can be expressed as:

$$m \geq 1\epsilon(\ln|H| + \ln 1\delta)$$

Where:

- mm is the sample size,
- ϵϵ is the maximum acceptable error,
- |H||H| is the size of the hypothesis space,

- δδ is the acceptable failure probability.

Importance of PAC Learning

Understanding PAC learning is essential for several reasons:

- **Theoretical Foundation**: It provides a rigorous foundation for analyzing the behavior and performance of learning algorithms, helping researchers and practitioners design better models.
- **Generalization Guarantees**: PAC learning offers theoretical guarantees regarding the generalization ability of algorithms, which is crucial for assessing their reliability.
- **Guidance for Sample Size**: By quantifying the sample complexity, PAC learning helps determine how much data is necessary for effective learning, which is particularly important in real-world applications.

Challenges in PAC Learning

Despite its advantages, PAC learning faces several challenges:

- **Computational Complexity**: Finding the optimal hypothesis can be computationally expensive, especially as the hypothesis space grows.
- **Model Assumptions**: PAC learning relies on certain assumptions about the underlying distribution of the data, which may not always hold in practice.
- **Overfitting**: As the complexity of the hypothesis space increases, there is a risk of overfitting, where the model performs well on training data but poorly on unseen data.

6. Explain in detail about Hypothesis spaces.

- The concept of a hypothesis is fundamental in Machine Learning and data science endeavours.
- In the realm of machine learning, a hypothesis serves as an initial assumption made by data scientists and ML professionals when attempting to address a problem.

- Machine learning involves conducting experiments based on past experiences, and these hypotheses are crucial in formulating potential solutions.
- It's important to note that in machine learning discussions, the terms "hypothesis" and "model" are sometimes used interchangeably.
- However, a hypothesis represents an assumption, while a model is a mathematical representation employed to test that hypothesis. This section on "Hypothesis in Machine Learning" explores key aspects related to hypotheses in machine learning and their significance.

Hypothesis in Machine Learning

- A hypothesis in machine learning is the model's presumption regarding the connection between the input features and the result.
- It is an illustration of the mapping function that the algorithm is attempting to discover using the training set.
- To minimize the discrepancy between the expected and actual outputs, the learning process involves modifying the weights that parameterize the hypothesis.
- The objective is to optimize the model's parameters to achieve the best predictive performance on new, unseen data, and a cost function is used to assess the hypothesis' accuracy.

How does a Hypothesis work?

- In most supervised machine learning algorithms, our main goal is to find a possible hypothesis from the hypothesis space that could map out the inputs to the proper outputs.
- The following fig.1.5 shows the common method to find out the possible hypothesis from the Hypothesis space:

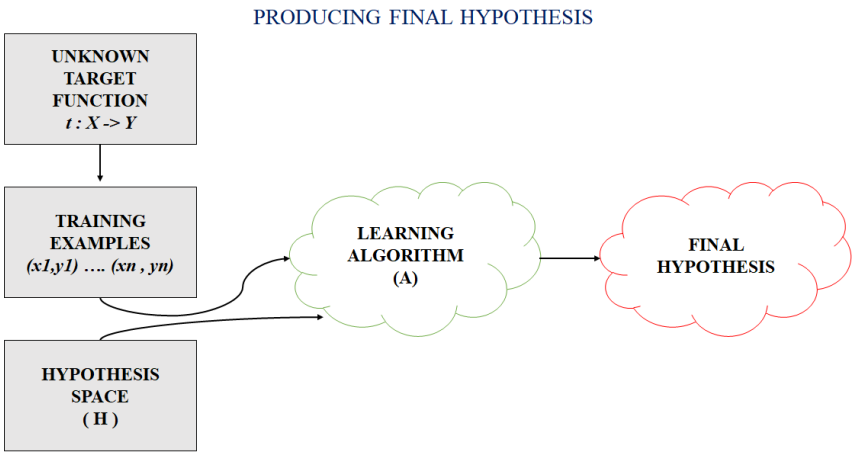


Fig.1.5 Producing Final Hypothesis

Hypothesis Space (H)

- Hypothesis space is the set of all the possible legal hypothesis. This is the set from which the machine learning algorithm would determine the best possible (only one) which would best describe the target function or the outputs.

Hypothesis (h)

- A hypothesis is a function that best describes the target in supervised machine learning. The hypothesis that an algorithm would come up depends upon the data and also depends upon the restrictions and bias that we have imposed on the data.

The Hypothesis can be calculated as:

$$y=mx+b$$

Where,

y = range

m = slope of the lines

x = domain

b = intercept

- To better understand the Hypothesis Space and Hypothesis consider the following coordinate that shows the distribution of some data in fig.1.6:

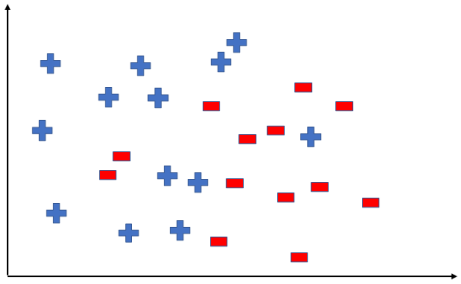


Fig.1.6 Distribution of Data

- Say suppose we have test data for which we have to determine the outputs or results. The test data is as shown below in fig.1.7:

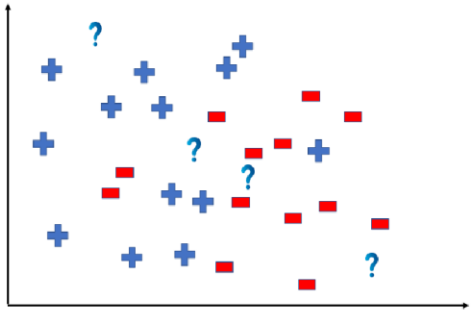


Fig.1.7 Test Data

- We can predict the outcomes by dividing the coordinate as shown below in fig.1.8:

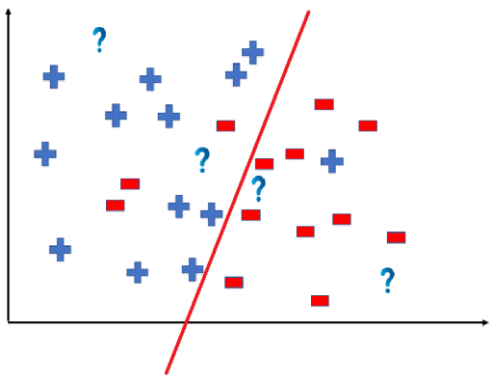


Fig.1.8 Predict the Outcome

- So the test data would yield the following result in fig. 1.9:

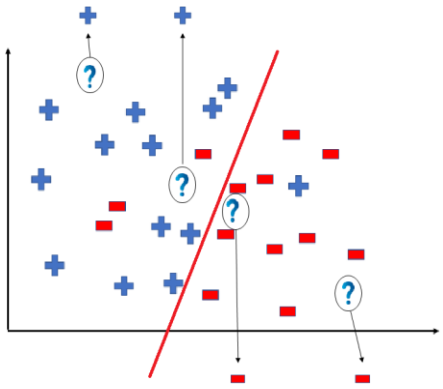


Fig.1.9 Result of Test Data

- But note here that we could have divided the coordinate plane as in fig. 1.10:

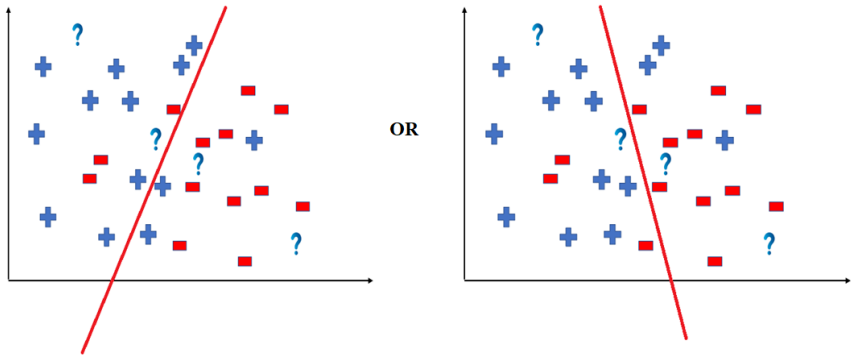


Fig.1.10 Divide the Coordinate plane

- The way in which the coordinate would be divided depends on the data, algorithm and constraints.
- All these legal possible ways in which we can divide the coordinate plane to predict the outcome of the test data composes of the Hypothesis Space.
- Each individual possible way is known as the hypothesis.
- Hence, in this example the hypothesis space would be like in fig.1.11:

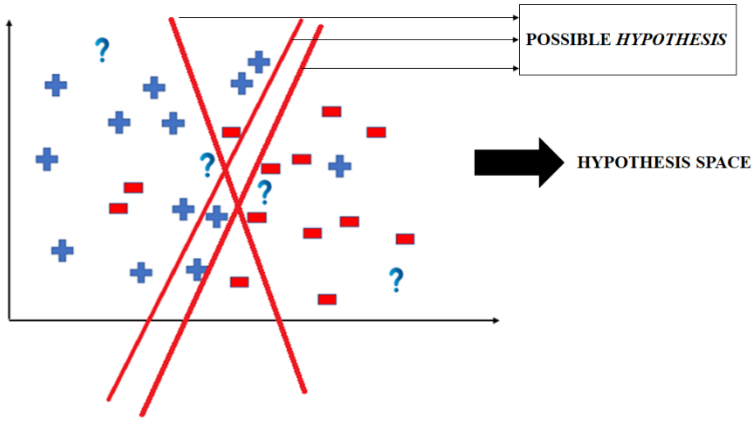


Fig.1.11 Hypothesis Space

Hypothesis Space and Representation in Machine Learning

- The hypothesis space comprises all possible legal hypotheses that a machine learning algorithm can consider.
- Hypotheses are formulated based on various algorithms and techniques, including linear regression, decision trees, and neural networks. These hypotheses capture the mapping function transforming input data into predictions.

Hypothesis Formulation and Representation in Machine Learning

- Hypotheses in machine learning are formulated based on various algorithms and techniques, each with its representation. For example:

  Linear Regression: $h(X) - \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \ldots + \theta_n X_n$
  Decision Trees: $h(X) - \text{Tree}(X)$
  Neural Networks: $h(X) - \text{NN}(X)$

- In the case of complex models like neural networks, the hypothesis may involve multiple layers of interconnected nodes, each performing a specific computation.

Hypothesis Evaluation:

- The process of machine learning involves not only formulating hypotheses but also evaluating their performance. This evaluation is typically done using a loss function or an evaluation metric that quantifies the disparity between predicted outputs and ground truth labels.

- Common evaluation metrics include mean squared error (MSE), accuracy, precision, recall, F1-score, and others. By comparing the predictions of the hypothesis with the actual outcomes on a validation or test dataset, one can assess the effectiveness of the model.

Hypothesis Testing and Generalization:

- Once a hypothesis is formulated and evaluated, the next step is to test its generalization capabilities.
- Generalization refers to the ability of a model to make accurate predictions on unseen data.
- A hypothesis that performs well on the training dataset but fails to generalize to new instances is said to suffer from overfitting. Conversely, a hypothesis that generalizes well to unseen data is deemed robust and reliable.
- The process of hypothesis formulation, evaluation, testing, and generalization is often iterative in nature. It involves refining the hypothesis based on insights gained from model performance, feature importance, and domain knowledge.
- Techniques such as hyperparameter tuning, feature engineering, and model selection play a crucial role in this iterative refinement process.

Hypothesis in Statistics

- In statistics, a hypothesis refers to a statement or assumption about a population parameter. It is a proposition or educated guess that helps guide statistical analyses. There are two types of hypotheses: the null hypothesis (H0) and the alternative hypothesis (H1 or Ha).
- **Null Hypothesis(H0):** This hypothesis suggests that there is no significant difference or effect, and any observed results are due to chance. It often represents the status quo or a baseline assumption.
- **Aternative Hypothesis(H1 or Ha):** This hypothesis contradicts the null hypothesis, proposing that there is a significant difference or effect in the population. It is what researchers aim to support with evidence.

7. Explain in detail about Inductive Bias.

Explain the Concept of Inductive Bias. [Nov/Dec 2024]

Write detailed notes on Inductive Bias and Bias Variance trade-off. [Apr/May 2024]

- In the realm of machine learning, the concept of inductive bias plays a pivotal role in shaping how algorithms learn from data and make predictions.
- It serves as a guiding principle that helps algorithms generalize from the training data to unseen data, ultimately influencing their performance and decision-making processes.
- In this article, we delve into the intricacies of inductive bias, its significance in machine learning, and its implications for model development and interpretation.

What is Inductive Bias?

- Inductive bias can be defined as the set of assumptions or biases that a learning algorithm employs to make predictions on unseen data based on its training data. These assumptions are inherent in the algorithm's design and serve as a foundation for learning and generalization.
- The inductive bias of an algorithm influences how it selects a hypothesis (a possible explanation or model) from the hypothesis space (the set of all possible hypotheses) that best fits the training data.
- It helps the algorithm navigate the trade-off between fitting the training data perfectly (overfitting) and generalizing well to unseen data (underfitting).

Types of Inductive Bias

Inductive bias can manifest in various forms, depending on the algorithm and its underlying assumptions. Some common types of inductive bias include:

- Bias towards simpler explanations: Many machine learning algorithms, such as decision trees and linear models, have a bias towards simpler hypotheses. They prefer explanations that are more parsimonious and less complex, as these are often more likely to generalize well to unseen data.
- Bias towards smoother functions: Algorithms like kernel methods or Gaussian processes have a bias towards smoother functions. They assume that neighboring

points in the input space should have similar outputs, leading to smooth decision boundaries.

- Bias towards specific types of functions: Neural networks, for example, have a bias towards learning complex, nonlinear functions. This bias allows them to capture intricate patterns in the data but can also lead to overfitting if not regularized properly.
- Bias towards sparsity: Some algorithms, like Lasso regression, have a bias towards sparsity. They prefer solutions where only a few features are relevant, which can improve interpretability and generalization.

Importance of Inductive Bias

- Inductive bias is crucial in machine learning as it helps algorithms generalize from limited training data to unseen data. Without a well-defined inductive bias, algorithms may struggle to make accurate predictions or may overfit the training data, leading to poor performance on new data.
- Understanding the inductive bias of an algorithm is essential for model selection, as different biases may be more suitable for different types of data or tasks. It also provides insights into how the algorithm is learning and what assumptions it is making about the data, which can aid in interpreting its predictions and results.

Challenges and Considerations

- While inductive bias is essential for learning, it can also introduce limitations and challenges. Biases that are too strong or inappropriate for the data can lead to poor generalization or biased predictions. Balancing bias with variance (the variability of predictions) is a key challenge in machine learning, requiring careful tuning and model selection.
- Additionally, the choice of inductive bias can impact the interpretability of the model. Simpler biases may lead to more interpretable models, while more complex biases may sacrifice interpretability for improved performance.

Conclusion

- Inductive bias is a fundamental concept in machine learning that shapes how algorithms learn and generalize from data. It serves as a guiding principle that

influences the selection of hypotheses and the generalization of models to unseen data.

- Understanding the inductive bias of an algorithm is essential for model development, selection, and interpretation, as it provides insights into how the algorithm is learning and making predictions. By carefully considering and balancing inductive bias, machine learning practitioners can develop models that generalize well and provide valuable insights into complex datasets.

8. Explain in detail about Generalization.

- Generalization is a measure of how your model performs on predicting unseen data. So, it is important to come up with the best-generalized model to give better performance against future data.
- Let us first understand what is underfitting and overfitting, and then see what are the best practices to train a generalized model in fig.1.12.

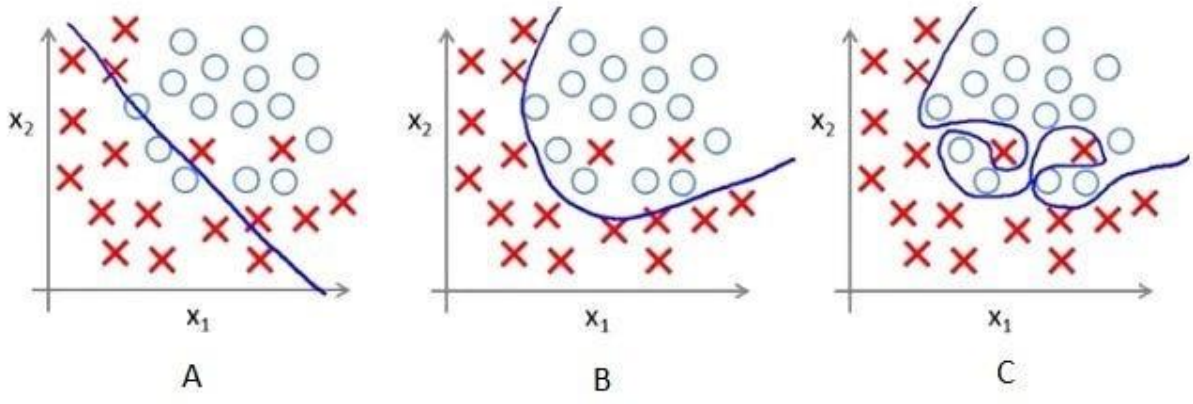


Fig.1.12 A: Underfitting, B: Generalized, C: Overfitting

What is Underfitting?

- Underfitting is a state where the model cannot model itself on the training data. And also not able to generalize new data. You can notice it with the help of loss function during your training. A simple rule of thumb is if both training loss and cross-validation loss are high, then your model is underfitting.

- Lack of data, not enough features, lack of variance in training data or high regularization rate can cause underfitting. A simple solution is to add more shuffled data to your training.
- Depending on what causes underfitting to your model, you can try introducing more meaningful features, feature crossing and introducing higher order polynomials as features or reducing regularization rate if you are using regularization. In some cases trying out with different training algorithm will work fine.

What is Overfitting?

- Overfitting is a situation where your model force learns the whole variance. Experts say it as model starts to memorize all the noise instead of learning. A simple rule of thumb to identify the overfitting is if your training loss is low and cross-validation loss is high then your model is overfitting.
- Uncleaned data, fewer steps in training, higher complexity of the model (due to higher weights in data) can cause overfitting. It is always recommended to preprocess data and create a good data pipeline.
- Select only necessary and meaningful features with good variance. Reduce the complexity of the model using good regularization algorithm (L1 norm or L2 norm) in fig.1.13.

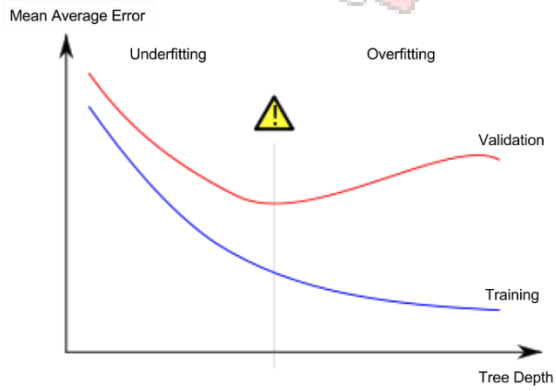


Fig.1.13 Overfitting

What are the best practices to get a Generalized model?

- It is important to have a training dataset with good variance (i.e. a shuffled data set). The Best way to do this is computing the hash for an appropriate feature and split data into training, evaluation and test sets based on the computed hash value. Here the evaluation set is used to cross-validate the trained model. It is always good to ensure that the distribution in all the dataset is stationary(same).
- Handling outliers also important, it always depends on the task you are working around. If you are training the model to detect anomalies you should consider outliers, in such case, these anomalies may be the labels you need to identify. So you cannot classify or detect without outliers. On the other hand, if you are modeling a regression-based classification it is good to remove outliers.
- Using resampling during the training. Resampling enables you to reconstruct your sample dataset in different ways for each iteration. One of the most popular resampling technique is k-fold cross-validation. It does training and testing on the model for k times with different subsets of your testing data.
- It is always good to know when to stop training. It is a common human insight to determine when to stop training. When you reach a good training loss and a good validation loss at that point stop training.
- Learn to do some feature engineering when needed. In some cases, your model cannot be able to converge, there may be not a meaningful relation found on the raw features you have. Doing Feature crosses and introducing new features with meaningful relation helps the model to converge.
- In addition to these parameters tunings, Hyper parameter tunings, using regularization algorithms also helps to generalize the model for better performance.
- Hope you all get a basic idea of generalization, underfitting, and overfitting. Use this as a base and keep exploring on subtopics for deeper understandings.
- Don't forget to applaud if you find this article useful. Your doubts and feedbacks are always welcomed.

9. Explain in detail about Bias variance trade-off.

Illustrate the tradeoff between the bias and variance in ML. [Nov/Dec 2024]

Write detailed notes on Inductive Bias and Bias Variance trade-off. [Apr/May 2024]

- It is important to understand prediction errors (bias and variance) when it comes to accuracy in any machine-learning algorithm. There is a tradeoff between a model's ability to minimize bias and variance which is referred to as the best solution for selecting a value of Regularization constant.
- A proper understanding of these errors would help to avoid the overfitting and underfitting of a data set while training the algorithm.

What is Bias?

- The bias is known as the difference between the prediction of the values by the Machine Learning model and the correct value. Being high in biasing gives a large error in training as well as testing data. It recommended that an algorithm should always be low-biased to avoid the problem of underfitting.
- By high bias, the data predicted is in a straight line format, thus not fitting accurately in the data in the data set. Such fitting is known as the Underfitting of Data. This happens when the hypothesis is too simple or linear in nature. Fig.1.14 Refer to the graph given below for an example of such a situation.

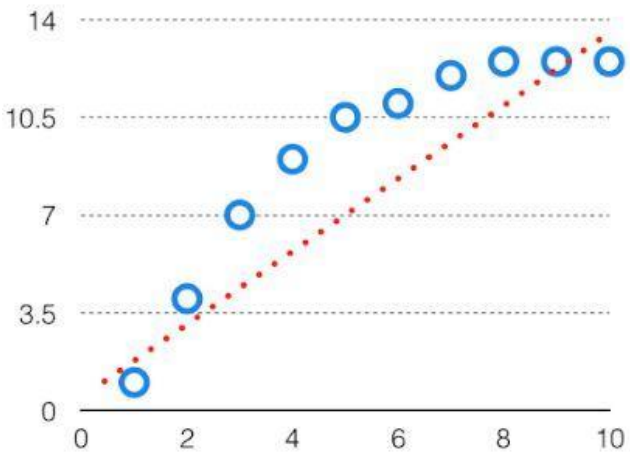


Fig.1.14 High Bias in the Model

In such a problem, a hypothesis looks like follows.

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

What is Variance?

- The variability of model prediction for a given data point which tells us the spread of our data is called the variance of the model. The model with high variance has a very

complex fit to the training data and thus is not able to fit accurately on the data which it hasn't seen before.

- As a result, such models perform very well on training data but have high error rates on test data. When a model is high on variance, it is then said to as Overfitting of Data.
- Overfitting is fitting the training set accurately via complex curve and high order hypothesis but is not the solution as the error with unseen data is high. While training a data model variance should be kept low. The high variance data looks as follows in fig.1.15.

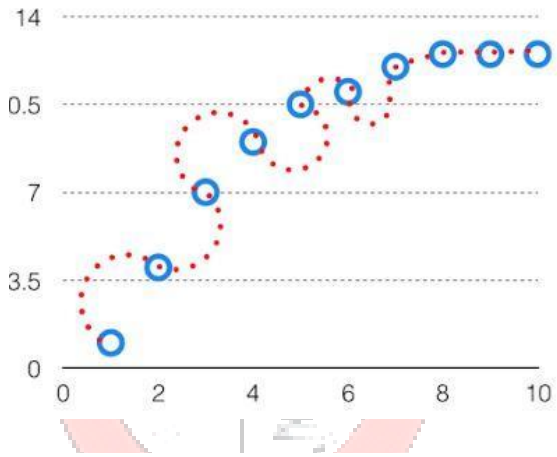


Fig.1.15 High Variance in the Model

In such a problem, a hypothesis looks like follows.

$$h_\theta(x) = g(\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4)$$

Bias Variance Tradeoff

- If the algorithm is too simple (hypothesis with linear equation) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree equation) then it may be on high variance and low bias. In the latter condition, the new entries will not perform well.
- Well, there is something between both of these conditions, known as a Trade-off or Bias Variance Trade-off. This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time. For the graph, the perfect tradeoff in fig.1.16.

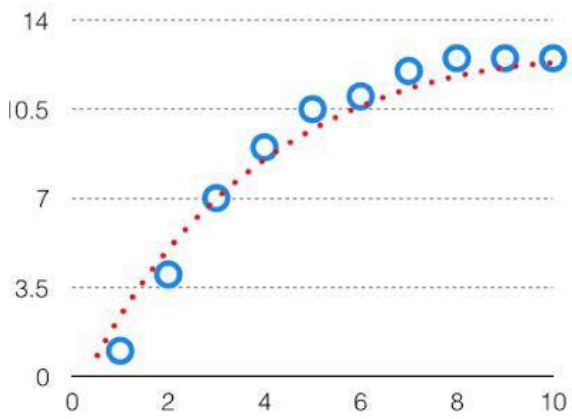


Fig.1.16 Bias Variance Tradeoff

- We try to optimize the value of the total error for the model by using the Bias-Variance Tradeoff.

$$\text{Total Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

- The best fit will be given by the hypothesis on the tradeoff point. The error to complexity graph to show trade-off is given as fig.1.17

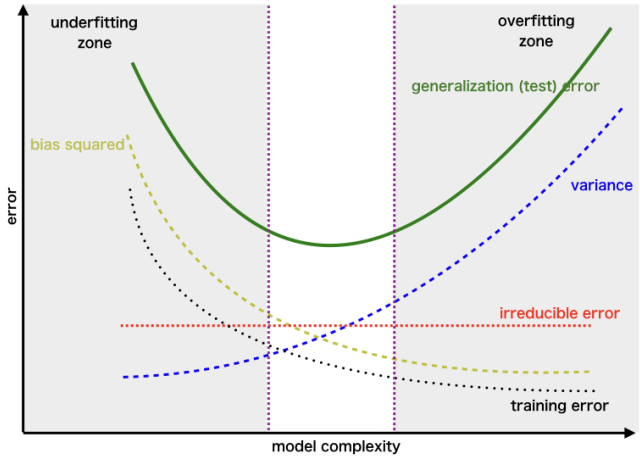


Fig.1.17 Region for the Least Value of Total Error

MAILAM Engineering College

(Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai,

Accredited by National Board of Accreditation (NBA), Accredited by NAAC with "A" Grade & (TCS), Chennai)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

II YEAR / IV SEM

AL3451- MACHINE LEARNING

UNIT II SUPERVISED LEARNING

Linear Regression Models: Least squares, single & multiple variables, Bayesian linear regression, gradient descent, Linear Classification Models: Discriminant function- Perceptron algorithm– Probabilistic discriminative model - Logistic regression, Probabilistic generative model – Naive Bayes, Maximum margin classifier – Support vector machine, Decision Tree, Random forests.

PART A

1. **Mention the merits of Bayesian linear regression. [Apr 2024]**

   - Extremely efficient when the dataset is tiny. Particularly well-suited for online learning as opposed to batch learning, when we know the complete dataset before we begin training the model.
   - This is so that Bayesian Regression can be used without having to save data.

2. **Distinguish between Random Forest and Support Vector Machine. [Apr 2024]**

   - A Random Forest is an ensemble learning method that builds multiple decision trees, combining their predictions to achieve a more robust result, while a Support Vector Machine (SVM) is a supervised learning algorithm that aims to find the optimal hyperplane to separate different classes in the data, excelling in high-dimensional spaces and handling small datasets effectively;
   - Essentially, Random Forest is better for interpretability and large datasets, while SVM shines in complex, high-dimensional data with smaller sample sizes.

3. **Differentiate between regression and classification. [Nov 2024]**

- Classification and regression are two primary tasks in supervised machine learning, where key difference lies in the nature of the output: classification deals with discrete outcomes (e.g., yes/no, categories), while regression handles continuous values (e.g., price, temperature).

4. What is support vector in SVM? [Nov 2024]

- A support vector is a data point that is closest to the decision boundary or hyperplane in a Support Vector Machine (SVM). Support vectors are important because they define the decision boundary and the margin of separation.

5. Define Regression.

- Regression is a supervised learning technique which helps in finding the correlation between variables and enables us to predict the continuous output variable based on the one or more predictor variables.
- It is mainly used for prediction, forecasting, time series modeling, and determining the causal-effect relationship between variables.

6. Define Classification.

- The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data.
- In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups.
- Such as, Yes or No, 0 or 1, Spam or Not Spam, cat or dog, etc. Classes can be called as targets/labels or categories.

7. Define Clustering.

- Clustering or cluster analysis is a machine learning technique, which groups the unlabeled dataset.
- It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points.
- The objects with the possible similarities remain in a group that has less or no similarities with another group."

8. What is Linear Regression?

- **In statistics, linear regression** is a linear approach to modeling the relationship between a dependent variable and one or more independent variables.
- Let **X** be the independent variable and **Y** be the dependent variable.
- A linear relationship between these two variables as follows:

$$Y = mX + c$$

Where,

m: Slope

c: y-intercept

9. What is Least Squares Regression Line?

- Least squares are a commonly used method in regression analysis for estimating the unknown parameters by creating a model which will minimize the sum of squared errors between the observed data and the predicted data.

Narrate Least Squares Regression Equation

- The equation that minimizes the total of all squared prediction errors for known Y scores in the original correlation analysis.

LEAST SQUARES REGRESSION EQUATION

$$Y' = bX + a$$

where

Y′ represents the predicted value;

X represents the known value;

b and a represent numbers calculated from the original correlation analysis

10. List and define the types of Linear Regression.

It is of two types: **Simple and Multiple**.

- **Simple Linear Regression** is where only one independent variable is present and the model has to find the linear relationship of it with the dependent variable
- **Equation of Simple Linear Regression,** where b_o is the intercept, b_1 is coefficient or slope, x is the independent variable and y is the dependent variable.

$$y = b_o + b_1 x$$

- In **Multiple Linear Regression** there are more than one independent variables for the model to find the relationship.

Equation of Multiple Linear Regression, where bo is the intercept, $b_1, b_2, b_3, b_4 \ldots, b_n$ are coefficients or slopes of the independent variables $x_1, x_2, x_3, x_4 \ldots, x_n$ and y is the dependent variable.

$$y = b_o + b_1 x_1 + b_2 x_2 + b_3 x_3 \ldots . + b_n x_n$$

11. Define Linear Regression Model.

- A Linear Regression model's main aim is to find the best fit linear line and the optimal values of intercept and coefficients such that the error is minimized.

12. What is error or residual?

- Error is the difference between the actual value and Predicted value and the goal is to reduce this difference.
- The vertical distance between the data point and the regression line is known as error or residual.
- Each data point has one residual and the sum of all the differences is known as **the Sum of Residuals/Errors.**

13. Define Bayesian Regression.

- Bayesian Regression is used when the data is insufficient in the dataset or the data is poorly distributed.
- The output of a Bayesian Regression model is obtained from a probability distribution.

- The aim of Bayesian Linear Regression is to find the '**posterior**' distribution for the model parameters.
- The expression for Posterior is :

$$Posterior = \frac{(Likelihood * Prior)}{Normalization}$$

where

- **Posterior:** It is the probability of an event to occur; say, H, given that another event; say, E has already occurred. i.e., P(H | E).
- Prior: It is the probability of an event H has occurred prior to another event. i.e., P(H)
- Likelihood: It is a likelihood function in which some parameter variable is marginalized.

14. List the Advantages and Disadvantages of Bayesian Regression.

Advantages of Bayesian Regression:

- Very effective when the size of the dataset is small.
- Particularly well-suited for on-line based learning (data is received in real-time), as compared to batch based learning, where we have the entire dataset on our hands before we start training the model. This is because Bayesian Regression doesn't need to store data.
- The Bayesian approach is a tried and tested approach and is very robust, mathematically. So, one can use this without having any extra prior knowledge about the dataset.

Disadvantages of Bayesian Regression:

- The inference of the model can be time-consuming.
- If there is a large amount of data available for our dataset, the Bayesian approach is not worth it.

15. What are the two types of Classifications problem?

- **Two-class problems:**
  - **Binary representation or Binary Classifier:**

- If the classification problem has only two possible outcomes, then it is called as Binary Classifier.
- There is a single target variable $t \in \{0, 1\}$
  - t = 1 represents class C1
  - t = 0 represents class C2
- **Examples:** YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.

- **Multi-class Problems:**
  - If a classification problem has more than two outcomes, then it is called as Multi-class Classifier.
  - **Example:** Classifications of types of crops, Classification of types of music.

16. List the different Types of ML Classification Algorithms.

- Logistic Regression
- K-Nearest Neighbors
- Support Vector Machines
- Kernel SVM
- Naïve Bayes
- Decision Tree Classification
- Random Forest Classification

17. What is Discriminant function?

- A function of a set of variables that is evaluated for samples of events or objects and used as an aid in discriminating between or classifying them.
- A discriminant function (DF) maps independent (discriminating) variables into a latent variable D.
- DF is usually postulated to be a linear function:

$$D = a_0 + a_1 x_1 + a_2 x_2 \ldots a_N x_N$$

18. Define Probabilistic discriminative models.

- **Discriminative models** are a class of supervised machine learning models which make predictions by estimating conditional probability P(y|x).
- For the two-class classification problem, the posterior probability of class C1 can be written as a logistic sigmoid acting on a linear function of x

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma\left(\ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}\right) = \sigma(\mathbf{w}^T\mathbf{x} + w_0)$$

- For the multi-class case, the posterior probability of class Ck is given by a softmax transformation of a linear function of x

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_{j=1}^{K} p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} = \frac{\exp\left(\mathbf{w}_k^T\mathbf{x} + w_{k0}\right)}{\sum_{j=1}^{K} \exp\left(\mathbf{w}_j^T\mathbf{x} + w_{j0}\right)}$$

19. Define Logistics Regression

- Logistic regression is the Machine Learning algorithms, under the classification algorithm of Supervised Learning technique.
- Logistic regression is used to describe data and the relationship between one dependent variable and one or more independent variables.
- The independent variables can be nominal, ordinal, or of interval type.
- Logistic regression predicts the output of a categorical dependent variable.
- Therefore the outcome must be a categorical or discrete value.

20. Define Logistic Function or Sigmoid Function.

- The logistic function is also known as the sigmoid function.
- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- The value of the logistic regression must be between 0 and 1, so it forms a curve like the "S" form.
- The S-form curve is called the Sigmoid function or the logistic function.

21. List the types of Logistic Regression.

- Logistic Regression can be classified into three types:

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

22. What are the Steps in Logistic Regression?

- To implement the Logistic Regression using Python, the steps are given below:
  - Data Pre-processing step
  - Fitting Logistic Regression to the Training set
  - Predicting the test result
  - Test accuracy of the result
  - Visualizing the test set result.

23. List the advantages of Logistic Regression Algorithm.

- Logistic regression performs better when the data is linearly separable
- It does not require too many computational resources
- There is no problem scaling the input features
- It is easy to implement and train a model using logistic regression

24. Define Probabilistic Generative model

- Given a model of one conditional probability, and estimated probability distributions for the variables X and Y, denoted P(X) and P(Y), can estimate the conditional probability using Bayes' rule:

$$P(X \mid Y)P(Y) = P(Y \mid X)P(X).$$

- A **generative model** is a statistical model of the joint probability distribution on given observable variable X and target variable Y.

Given a generative model for P(X|Y), can estimate:

$$P(Y \mid X) = P(X \mid Y)P(Y)/P(X),$$

25. Define Discriminative model.

- A **discriminative model** is a model of the conditional probability of the target Y, given an observation x given a discriminative model for P(Y|X), can estimate:

$$P(X \mid Y) = P(Y \mid X)P(X)/P(Y)$$

- Classifier based on a generative model is a **generative classifier**, while a classifier based on a discriminative model is a **discriminative classifier**

26. List the types of Generative models.

- Types of generative models are:
  - Naive Bayes classifier or Bayesian network
  - Linear discriminant analysis

27. Mention the algorithms in Discriminative models.

- Logistic regression
- Support Vector Machines
- Decision Tree Learning
- Random Forest

28. Define Support Vector Machine (SVM)

- Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression.
- The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points.
- Hyperplanes are decision boundaries that help classify the data points.

29. Define Hinge loss function

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

$$c(x, y, f(x)) = (1 - y * f(x))_+$$

- The cost is 0 if the predicted value and the actual value are of the same sign. If they are not, then calculate the loss value.

30. Define SVM Kernel.

- The SVM kernel is a function that takes low dimensional input space and transforms it into higher-dimensional space, ie it converts non-separable problem to separable problem. It is mostly useful in non-linear separation problems.

31. List the types of SVMs

- There are two different types of SVMs, each used for different things:
  - **Simple SVM:** Typically used for linear regression and classification problems.
  - **Kernel SVM**: Has more flexibility for non-linear data.

32. What are the advantages and disadvantages of SVM?

Advantages

- Effective on datasets with multiple features, like financial or medical data.
- Effective in cases where number of features is greater than the number of data points.
- Its memory efficient as it uses a subset of training points in the decision function called support vectors
- Different kernel functions can be specified for the decision functions and its possible to specify custom kernels

Disadvantages

- If the number of features is a lot bigger than the number of data points, choosing kernel functions and regularization term is crucial.
- SVMs don't directly provide probability estimates. Those are calculated using an expensive five-fold cross-validation.
- Works best on small sample sets because of its high training time.

33. List the applications of SVM.

SVMs can be used to solve various real-world problems:

- SVMs are helpful in text and hypertext categorization.
- Classification of images can also be performed using SVMs.
- Classification of satellite data like SAR data using supervised SVM.
- Hand-written characters can be recognized using SVM.
- The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90% of the compounds classified correctly.

34. Define Decision Tree

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems.
- It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, the Decision Node and Leaf Node.

35. List the types of Decision Trees

1. **Categorical Variable Decision Tree:** Decision Tree which has a categorical target variable then it called a **Categorical variable decision tree.**
2. **Continuous Variable Decision Tree:** Decision Tree has a continuous target variable then it is called **Continuous Variable Decision Tree.**

36. Mention the reason for using Decision Trees

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

37. List the algorithms used to construct Decision Trees:

- ID3 → (extension of D3)
- C4.5 → (successor of ID3)

- CART → (Classification And Regression Tree)
- CHAID → (Chi-square automatic interaction detection Performs multi-level splits when computing classification trees)
- MARS → (multivariate adaptive regression splines)

PART B

1. **Explain in detail about Linear Regression Models. Or Explain Linear Regression Models: Least squares, single & multiple variables, Bayesian linear regression, gradient descent.**

1.1 Linear Regression

- In statistics, linear regression is a linear approach to modeling the relationship between a dependent variable and one or more independent variables.
- Let **X** be the independent variable and **Y** be the dependent variable.
- A linear relationship between these two variables as follows:

$$Y = mX + c$$

Where,

m: Slope

c: y-intercept

- Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression.
- Linear regression finds how the value of the dependent variable is changing according to the value of the independent variable.
- The linear regression model provides a sloped straight line representing the relationship between the variables.
- Consider the below Figure 2.1, which represents the relationship between independent and dependent variables

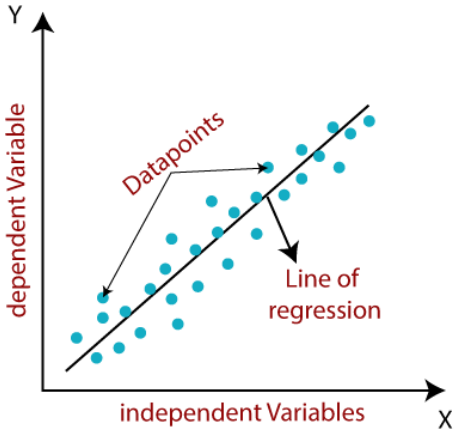


Figure 2.1 – Relationship between independent and dependent variables

1.2 Least Squares Regression Line

- Least squares are a commonly used method in regression analysis for estimating the unknown parameters by creating a model which will minimize the sum of squared errors between the observed data and the predicted data.

1.2.1 Least Squares Regression Equation

- The equation that minimizes the total of all squared prediction errors for known Y scores in the original correlation analysis.

LEAST SQUARES REGRESSION EQUATION

$$Y' = bX + a$$

where

Y´ represents the predicted value;

X represents the known value;

b and a represent numbers calculated from the original correlation analysis

1.2.2 Least Squares Regression in Python

Scenario: A rocket motor is manufactured by combining an igniter propellant and a sustainer propellant inside a strong metal housing. It was noticed that the shear strength of the bond between two

propellers is strongly dependent on the age of the sustainer propellant.

Problem Statement: Implement a simple linear regression algorithm using Python to build a machine learning model that studies the relationship between the shear strength of the bond between two propellers and the age of the sustainer propellant.

Step 1: Import the required Python libraries.

```
# Importing Libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt
```

Step 2: Next step is to read and load the dataset.

```
# Loading dataset

data = pd.read_csv('PropallantAge.csv')

data.head()

data.info()
```

Step 3: Create a scatter plot just to check the relationship between these two variables.

```
# Plotting the data

plt.scatter(data['Age of Propellant'],data['Shear Strength'])
```

Step 4: Next step is to assign X and Y as independent and dependent variables respectively.

```
# Computing X and Y

X = data['Age of Propellant'].values

Y = data['Shear Strength'].values
```

Step 5: Compute the mean of variables X and Y to determine the values of slope (m) and y-intercept.

Also, let n be the total number of data points.

```
# Mean of variables X and Y
```

```
mean_x = np.mean(X)
mean_y = np.mean(Y)
# Total number of data values
n = len(X)
```

Step 6: Calculate the slope and the y-intercept using the formulas

```
# Calculating 'm' and 'c'
num = 0
denom = 0
for i in range(n):
num += (X[i] - mean_x) * (Y[i] - mean_y)
denom += (X[i] - mean_x) ** 2
m = num / denom
c = mean_y - (m * mean_x)
# Printing coefficients
print("Coefficients")
print(m, c)
```

The above step has given the values of m and c. Substituting them ,

Shear Strength =

2627.822359001296 + (-37.15359094490524)

*** Age of Propellant**

Step 7: The above equation represents the linear regression model.

Let's plot this graphically. Refer fig 2.2

```
# Plotting Values and Regression Line
maxx_x = np.max(X) + 10
minn_x = np.min(X) - 10
# line values for x and y
x = np.linspace(minn_x, maxx_x, 1000)
```

```
y = c + m * x

# Plotting Regression Line

plt.plot(x, y, color='#58b970', label='Regression Line')

# Plotting Scatter Points

    plt.scatter(X, Y, c='#ef5423', label='Scatter Plot')

    plt.xlabel('Age of Propellant (in years)')

    plt.ylabel('Shear Strength')

    plt.legend()

    plt.show()
```

Output:

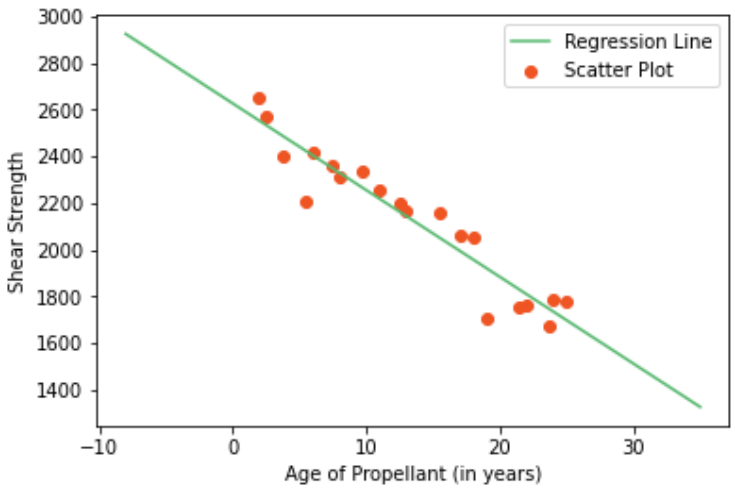


Figure 2.2 – Example for Regression Line

1.3 Types of Linear Regression

It is of two types: **Simple and Multiple**.

- **Simple Linear Regression** is where only one independent variable is present and the model has to find the linear relationship of it with the dependent variable

  Equation of Simple Linear Regression, where b_o is the intercept, b_1 is coefficient or slope, x is the independent variable and y is the dependent variable.

$$y = b_o + b_1 x$$

- In **Multiple Linear Regression** there are more than one independent variables for the model to find the relationship.

  Equation of Multiple Linear Regression, where bo is the intercept, $b_1, b_2, b_3, b_4 \ldots, b_n$ are coefficients or slopes of the independent variables $x_1, x_2, x_3, x_4 \ldots, x_n$ and y is the dependent variable.

$$y = b_o + b_1 x_1 + b_2 x_2 + b_3 x_3 \ldots . + b_n x_n$$

1.4 Linear Regression Model

- A Linear Regression model's main aim is to find the best fit linear line and the optimal values of intercept and coefficients such that the error is minimized.
- Error is the difference between the actual value and Predicted value and the goal is to reduce this difference.

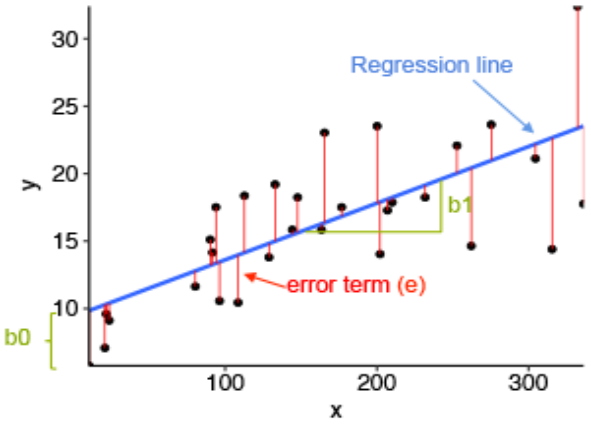


Figure 2.3 – Example for Linear Regression Model

- In the above figure 2.3,
  - x is our dependent variable which is plotted on the x-axis and y is the dependent variable which is plotted on the y-axis.
  - Black dots are the data points i.e the actual values.
  - b_o is the intercept which is 10 and b_1 is the slope of the x variable.

- The blue line is the best fit line predicted by the model i.e the predicted values lie on the blue line.

- The vertical distance between the data point and the regression line is known as error or residual.
- Each data point has one residual and the sum of all the differences is known as the Sum of Residuals/Errors.

Mathematical Approach:

Residual/Error = Actual values – Predicted Values

Sum of Residuals/Errors = Sum(Actual- Predicted Values)

Square of Sum of Residuals/Errors = (Sum(Actual- Predicted Values))^2

$$\sum e_i^2 = \sum (Y_i - \hat{Y}_i)^2$$

1.5 Bayesian Regression

- Bayesian Regression is used when the data is insufficient in the dataset or the data is poorly distributed.
- The output of a Bayesian Regression model is obtained from a probability distribution.
  - The aim of Bayesian Linear Regression is to find the '**posterior**' distribution for the model parameters.
  - The expression for Posterior is :

$$Posterior = \frac{(Likelihood * Prior)}{Normalization}$$

where

- **Posterior:** It is the probability of an event to occur; say, H, given that another event; say, E has already occurred. i.e., P(H | E).
- Prior: It is the probability of an event H has occurred prior to another event. i.e., P(H)
- Likelihood: It is a likelihood function in which some parameter variable is marginalized.

- The Bayesian Ridge Regression formula is as follows:

$$p(y|\lambda) = N(w|0, \lambda^{-1}Ip)$$

where

- 'y' is the expected value,
- lambda is the distribution's shape parameter before the lambda parameter
- the vector "w" is made up of the elements w0, w1,....

1.5.1 Implementation of Bayesian Regression Using Python

- Boston Housing dataset, which includes details on the average price of homes in various Boston neighborhoods.
- The r2 score will be used for evaluation.
- The crucial components of a Bayesian Ridge Regression model:

Program

```
fromsklearn.datasets import load_boston
fromsklearn.model_selection import train_test_split
fromsklearn.metrics import r2_score
fromsklearn.linear_model import BayesianRidge
# Loading the dataset
        dataset = load_boston()
        X, y = dataset.data, dataset.target
# Splitting the dataset into testing and training sets
        X_train, X_test, y_train, y_test = train_test_split
                (X, y, test_size = 0.15, random_state = 42)
# Creating to train the model
        model = BayesianRidge()
        model.fit(X_train, y_train)
# Model predicting the test data
        prediction = model.predict(X_test)
```

 # Evaluation of r2 score of the model against the test dataset

```
print(f"Test Set r2 score : {r2_score(y_test, prediction)}")
```

Output

```
Test Set r2 score : 0.7943355984883815
```

Advantages of Bayesian Regression:

- Very effective when the size of the dataset is small.
- Particularly well-suited for on-line based learning (data is received in real-time), as compared to batch based learning, where we have the entire dataset on our hands before we start training the model. This is because Bayesian Regression doesn't need to store data.
- The Bayesian approach is a tried and tested approach and is very robust, mathematically. So, one can use this without having any extra prior knowledge about the dataset.

Disadvantages of Bayesian Regression:

- The inference of the model can be time-consuming.
- If there is a large amount of data available for our dataset, the Bayesian approach is not worth it.

1.6 Gradient descent

1.6.1 Cost Function

- The cost is the error in our predicted value.
- It is calculated using the Mean Squared Error function as shown in figure 2.4.

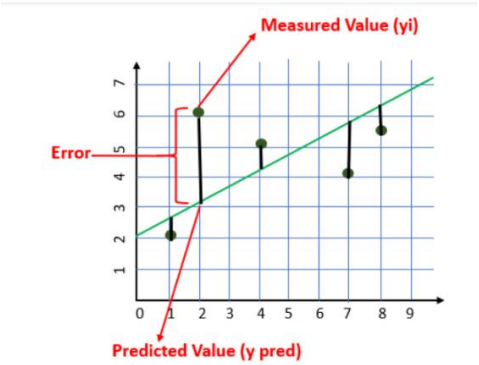


Figure 2.4 – Example for Cost function

$$Cost\ Function(MSE) = \frac{1}{n}\sum_{i=0}^{n}(y_i - y_{i\,pred})^2$$

Replace $y_{i\,pred}$ with $mx_i + c$

$$Cost\ Function(MSE) = \frac{1}{n}\sum_{i=0}^{n}(y_i - (mx_i + c))^2$$

- The goal is to minimize the cost as much as possible in order to find the best fit line.

1.6.2 Gradient Descent Algorithm.

- Gradient descent is an optimization algorithm that finds the best-fit line for a given training dataset in a smaller number of iterations.
- If m and c are plotted against MSE, it will acquire a bowl shape as shown in figure 2.4a and figure 2.4b.

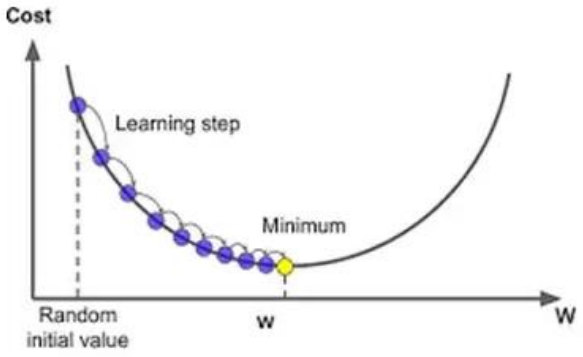


Figure 2.4a – Process of gradient descent algorithm

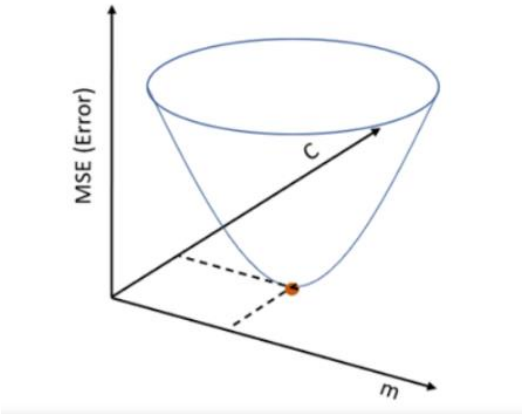


Figure 2.4b – Gradient Descent Shape

Learning Rate

- A learning rate is used for each pair of input and output values. It is a scalar factor and coefficients are updated in direction towards minimizing error.
- The process is repeated until a minimum sum squared error is achieved or no further improvement is possible.

Step by Step Algorithm:

1. Initially, let m = 0, c = 0

   Where L = learning rate — controlling how much the value of "m" changes with each step.

   The smaller the L, greater the accuracy. L = 0.001 for a good accuracy.

1. Calculating the partial derivative of loss function "m" to get the derivative D.

$$D_m = \frac{\partial(Cost\ Function)}{\partial m} = \frac{\partial}{\partial m}\left(\frac{1}{n}\sum_{i=0}^{n}(y_i - y_{i\,pred})^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial m}\left(\sum_{i=0}^{n}(y_i - (mx_i + c))^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial m}\left(\sum_{i=0}^{n}(y_i^2 + m^2x_i^2 + c^2 + 2mx_ic - 2y_imx_i - 2y_ic)\right)$$

$$= \frac{-2}{n}\sum_{i=0}^{n}x_i(y_i - (mx_i + c))$$

$$= \frac{-2}{n}\sum_{i=0}^{n}x_i(y_i - y_{i\,pred})$$

- Similarly, find the partial derivative with respect to c, Dc..

$$D_c = \frac{\partial(Cost\ Function)}{\partial c} = \frac{\partial}{\partial c}\left(\frac{1}{n}\sum_{i=0}^{n}(y_i - y_{i\,pred})^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial c}\left(\sum_{i=0}^{n}(y_i - (mx_i + c))^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial c}\left(\sum_{i=0}^{n}(y_i^2 + m^2x_i^2 + c^2 + 2mx_ic - 2y_imx_i - 2y_ic)\right)$$

$$= \frac{-2}{n}\sum_{i=0}^{n}(y_i - (mx_i + c))$$

$$\frac{-2}{n}\sum_{i=0}^{n}(y_i - y_{i\,pred})$$

- Update the current values of m and c using the following equation:

$$m = m - LD_m$$

$$c = c - LDc$$

- Repeat this process until our Cost function is very small (ideally 0).

2. Explain in detail about Linear Classification Models – Discriminant function.

2.1 Linear Classification Models

- The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data.
- In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, **Yes or No, 0 or 1, Spam or Not Spam, cat or dog,** etc.
- Classes can be called as targets/labels or categories.

- The output variable of Classification is a category, not a value, such as "Green or Blue", "fruit or animal", etc.
- Since the Classification algorithm is a supervised learning technique, hence it takes labeled input data, which means it contains input with the corresponding output.
- In classification algorithm, a discrete output function(y) is mapped to input variable(x).

$$y=f(x), \text{ where } y = \text{categorical output}$$

- The best example of an ML classification algorithm is **Email Spam Detector**.
- The goal of the classification algorithm is
    - Take a D-dimensional input vector x
    - Assign it to one of K discrete classes Ck , k = 1, . . . , K
- In the most common scenario, the classes are taken to be disjoint and each input is assigned to one and only one class
- The input space is divided into decision regions
- The boundaries of the decision regions
    - decision boundaries
    - decision surfaces
- With linear models for classification, the decision surfaces are linear functions, Classes that can be separated well by linear surfaces are linearly separable.
- In the figure 2.5, there are two classes, class A and Class B.
- These classes have features that are similar to each other and dissimilar to other classes.

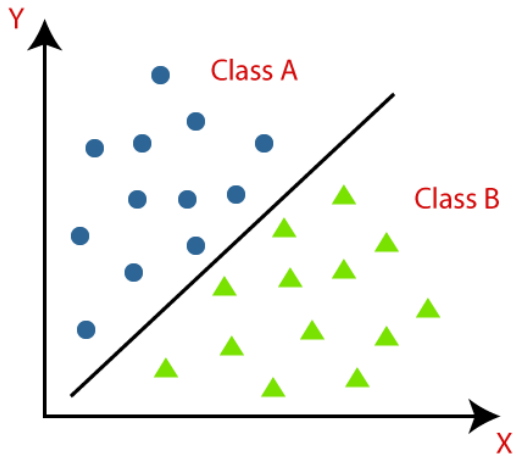


Figure 2.5 – Example of Classification

- The algorithm which implements the classification on a dataset is known as a classifier.
- There are two types of Classifications:
  - **Two-class problems:**
    - **Binary representation or Binary Classifier:**
      - If the classification problem has only two possible outcomes, then it is called as Binary Classifier.
      - There is a single target variable t ∈ {0, 1}
        - t = 1 represents class C1
        - t = 0 represents class C2
      - **Examples:** YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.
  - **Multi-class Problems:**
    - If a classification problem has more than two outcomes, then it is called as Multi-class Classifier.
    - **Example:** Classifications of types of crops, Classification of types of music.
  - **1-of-K coding scheme**
    - There is a K-long target vector t, such that

If the class is Cj, all elements t_k of t are zero for k ≠ j and one for k = j t_k is the probability that the class is C_k, K = 6 and C_k = 4, then t = $(0, 0, 0, 1, 0, 0)^T$

- The simplest approach to classification problems is through construction of a discriminant function that directly assigns each vector x to a specific class

2.2 Types of ML Classification Algorithms:

- Logistic Regression
- K-Nearest Neighbors
- Support Vector Machines
- Kernel SVM
- Naïve Bayes
- Decision Tree Classification
- Random Forest Classification

2.3 Discriminant function

- A function of a set of variables that is evaluated for samples of events or objects and used as an aid in discriminating between or classifying them.
- A discriminant function (DF) maps independent (discriminating) variables into a latent variable D.
- DF is usually postulated to be a linear function:

$$D = a_0 + a_1 x_1 + a_2 x_2 \ldots a_N x_N$$

- The goal of discriminant analysis is to find such values of the coefficients $\{a_i, i=0,\ldots,N\}$ that the distance between the mean values of DF is maximal for the two groups.
- Whenever there is a requirement to separate two or more classes having multiple features efficiently, the Linear Discriminant Analysis model is considered the most common technique to solve such classification problems.

- For example, if there are classes with multiple features and need to separate them efficiently. Classify them using a single feature, then it may show overlapping as shown in figure 3.6.

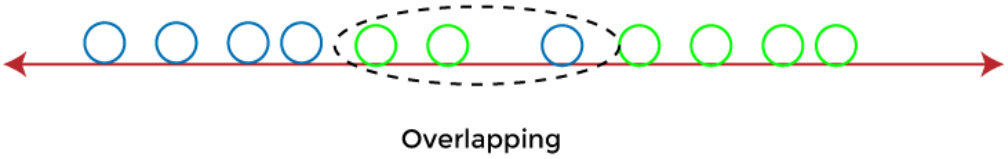


Figure 2.6 – Example for Classification using single feature

- To overcome the overlapping issue in the classification process, must increase the number of features regularly.

3. Explain in detail about Linear Discriminant Functions and its types. Also elaborate about logistic regression in detail.

LINEAR DISCRIMINANT FUNCTIONS AND LOGISTIC REGRESSION

3.1 Linear Discriminant Functions
3.2 The Two-Category Case
3.3 The Multi-category Case
3.4 Generalized Linear Discriminant Functions
3.5 Probabilistic discriminative models
3.6 Logistics Regression
3.6.1 Logistic Function (Sigmoid Function)
3.6.2 Assumptions for Logistic Regression
3.6.3 Logistic Regression Equation
3.6.4 Type of Logistic Regression
3.6.5 Steps in Logistic Regression
3.6.6 Advantages of Logistic Regression Algorithm

3.1 Linear Discriminant Functions

A discriminant function that is a linear combination of the components of x can be written as

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 \qquad (2.1)$$

where **w** is the weight vector and w_0 the bias or threshold weight.

3.2 The Two-Category Case

- For a discriminant function of the form of eq.2.1, a two-category classifier implements the following decision rule:
- Decide w_1 if $g(\mathbf{x})>0$ and w_2 if $g(\mathbf{x})<0$.
- Thus, $\mathbf{x}$ is assigned to w_1 if the inner product $\mathbf{w}^T\mathbf{x}$ exceeds the threshold – w_0 and to w_2 otherwise.
- If $g(\mathbf{x})=0$, $\mathbf{x}$ can ordinarily be assigned to either class, or can be left undefined.
- The equation $g(\mathbf{x})=0$ defines the decision surface that separates points assigned to w_1 from points assigned to w_2.
- When g(x) is linear, this decision surface is a hyperplane.
- If $\mathbf{x}_1$ and $\mathbf{x}_2$ are both on the decision surface, then

$$\mathbf{w}^T\mathbf{x}_1 + w_0 = \mathbf{w}^T\mathbf{x}_2 + w_0 \qquad (2.2)$$

or

$$\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = 0 \qquad (2.3)$$

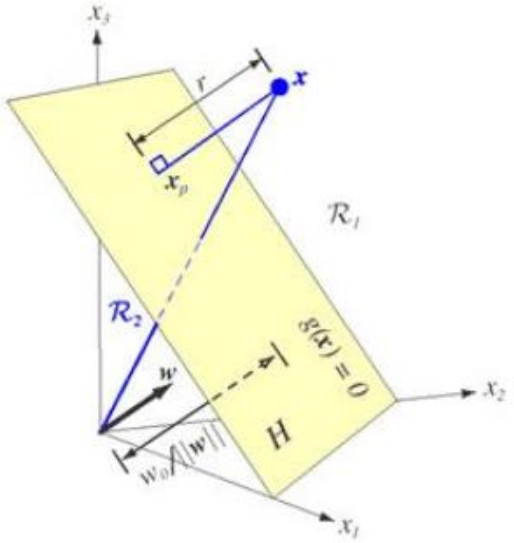


Figure 2.7: The linear decision boundary H separates the feature space into two half-spaces.

- In figure 2.7, the hyperplane H divides the feature space into two half-spaces:

- Decisionregion R_1 for w_1
- region R_2 for w_2.

- The discriminant function g(**x**) gives an algebraic measure of the distance from **x** to the hyperplane.
- The way to express **x** as

$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|} \qquad (2.4)$$

where $\mathbf{x}_p$ is the normal projection of **x** onto H, and r is the desired algebraic distance which is positive if **x** is on the positive side and negative if **x** is on the negative side. Then, because $g(\mathbf{x}_p)=0$,

$$\mathbf{x}_p = \mathbf{x} - r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$g(\mathbf{x}_p) = \mathbf{w}^T\mathbf{x}_p + w_0 = 0$$

$$g(\mathbf{x}_p) = \mathbf{w}^T\left(\mathbf{x} - r\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) + w_0 = 0$$

$$\mathbf{w}^T\mathbf{x} - r\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|}) + w_0 = 0$$

Since $\mathbf{w}^T\mathbf{w} = \|\mathbf{w}\|^2$ then

$$\mathbf{w}^T\mathbf{x} + w_0 - r\|\mathbf{w}\| = 0$$

$$\mathbf{w}^T\mathbf{x} + w_0 = r\|\mathbf{w}\|$$

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = r\|\mathbf{w}\| \qquad (2.5)$$

or

$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|} \qquad (2.6)$$

- The distance from the origin to H is given by $\frac{w_0}{\|\mathbf{w}\|}$.

- If $w_0>0$, the origin is on the positive side of H, and if $w_0<0$, it is on the negative side.
- If $w_0=0$, then g(**x**) has the homogeneous form $\mathbf{w}^T\mathbf{x}$, and the hyperplane passes through the origin

3.3 The Multi-category Case

- To devise multi category classifiers employing linear discriminant functions reduce the problem to c two-class problems.
- Defining c linear discriminant functions

$$g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0} \qquad i = 1,\ldots,c \tag{2.7}$$

and assigning **x** to w_i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all j^1 i; in case of ties, the classification is left undefined.

- The resulting classifier is called a linear machine.

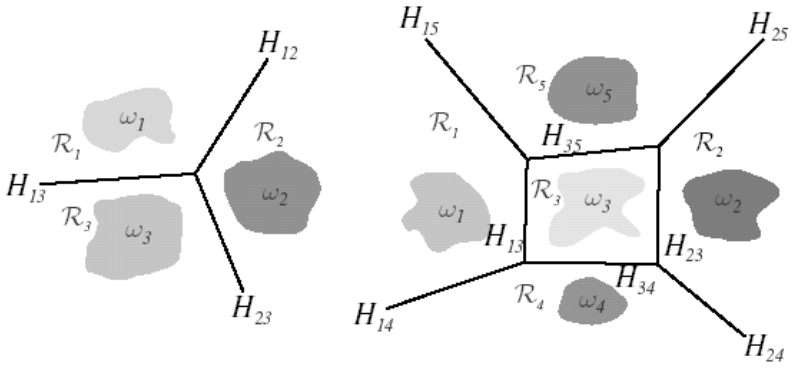


Figure 2.8: Decision boundaries defined by linear machines

- A linear machine divides the feature space into c decision regions as shown in figure 2.8, with g_j(**x**) being the largest discriminant if **x** is in region R_i.
- If R_i and R_j are contiguous, the boundary between them is a portion of the hyperplane H_{ij} defined by

$$g_i(\mathbf{x}) = g_j(\mathbf{x}) \text{ or } \qquad (\mathbf{w}_i - \mathbf{w}_j)^T\mathbf{x} + (w_{i0} - w_{j0}) = 0$$

$\mathbf{w}_i - \mathbf{w}_j$ is normal to H_{ij} and the signed distance from $\mathbf{x}$ to H_{ij} is given by

$$\frac{(g_i(\mathbf{x}) - g_j(\mathbf{x}))}{\|\mathbf{w}_i - \mathbf{w}_j\|}$$

3.4 Generalized Linear Discriminant Functions

- The linear discriminant function g(**x**) can be written as

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i \qquad (2.8)$$

where the coefficients w_i are the components of the weight vector **w**.

Quadratic Discriminant Function

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j \qquad (2.9)$$

3.5 Probabilistic discriminative models

- **Discriminative models** are a class of supervised machine learning models which make predictions by estimating conditional probability P(y | x).
- For the two-class classification problem, the posterior probability of classC1 can be written as a logistic sigmoid acting on a linear function of x

$$p(C_1|\mathbf{x}) = \sigma\left(\ln \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)}\right) = \sigma(\mathbf{w}^T\mathbf{x} + w_0)$$

- For the multi-class case, the posterior probability of class Ckis given by a softmax transformation of a linear function of x

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_{j=1}^{K} p(\mathbf{x}|C_j)p(C_j)} = \frac{\exp(\mathbf{w}_k^T\mathbf{x} + w_{k0})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^T\mathbf{x} + w_{j0})}$$

3.6 Logistics Regression

- Logistic regression is the Machine Learning algorithms, under the classification algorithm of Supervised Learning technique.
- Logistic regression is used to describe data and the relationship between one dependent variable and one or more independent variables.
- The independent variables can be nominal, ordinal, or of interval type.
- Logistic regression predicts the output of a categorical dependent variable.
- Therefore the outcome must be a categorical or discrete value.
- It can be either Yes or No, 0 or 1, true or False, etc. it gives the probabilistic values which lie between 0 and 1.
- Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.
- The figure 2.9 predicts the logistic function

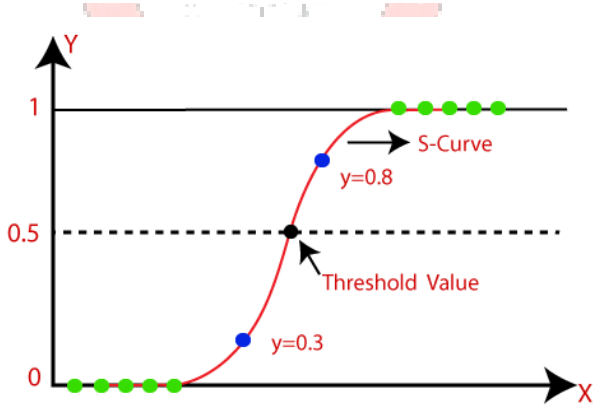


Figure 2.9 – Logistic Function or Sigmoid Function

3.6.1 Logistic Function (Sigmoid Function):

- The logistic function is also known as the sigmoid function.
- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- The value of the logistic regression must be between 0 and 1, so it forms a curve like the "S" form.

- The S-form curve is called the Sigmoid function or the logistic function.

3.6.2 Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

3.6.3 Logistic Regression Equation:

- The Logistic regression equation can be obtained from the Linear Regression equation.
- The mathematical steps are given below:
  - The equation of the straight line can be written as:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

  - In Logistic Regression y can be between 0 and 1 only, let's divide the above equation by (1-y):

$$\frac{y}{1-y}\text{; 0 for y= 0, and infinity for y=1}$$

  - For the range between -[infinity] to +[infinity], take logarithm of the equation:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

3.6.4 Type of Logistic Regression:

- Logistic Regression can be classified into three types:
  - **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
  - **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"

- **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

3.6.5 Steps in Logistic Regression:

- To implement the Logistic Regression using Python, the steps are given below:
  - Data Pre-processing step
  - Fitting Logistic Regression to the Training set
  - Predicting the test result
  - Test accuracy of the result
  - Visualizing the test set result.

3.6.6 Advantages of Logistic Regression Algorithm

- Logistic regression performs better when the data is linearly separable
- It does not require too many computational resources
- There is no problem scaling the input features
- It is easy to implement and train a model using logistic regression

4. Elaborate in detail about Probabilistic Generative model and Naïve Bayes.

PROBABILISTIC GENERATIVE MODEL AND NAÏVE BAYES

4.1 Probabilistic Generative model

4.2 Simple example

4.3 Generative models

4.4 Discriminative models

4.1 Probabilistic Generative model

- Given a model of one conditional probability, and estimated probability distributions for the variables X and Y, denoted P(X) and P(Y), can estimate the conditional probability using Bayes' rule:

$$P(X \mid Y)P(Y) = P(Y \mid X)P(X).$$

- A **generative model** isa statistical model of the joint probability distribution on given observable variable X and target variable Y.

  Given a generative model for P(X|Y), can estimate:

$$P(Y \mid X) = P(X \mid Y)P(Y)/P(X).$$

- A **discriminative model** is a model of the conditional probability of the target Y, given an observation xgiven a discriminative model forP(Y|X), can estimate:

$$P(X \mid Y) = P(Y \mid X)P(X)/P(Y).$$

- Classifier based on a generative model is a **generative classifier**, while a classifier based on a discriminative model is a **discriminative classifier**

4.2 Simple example

Suppose the input data is $x \in \{1, 2\}$, the set of labels for x is $y \in \{0, 1\}$, and there are the following 4 data points: $(x, y) = \{(1, 0), (1, 1), (2, 0), (2, 0)\}$

For the above data, estimating the joint probability distribution $p(x, y)$ from the empirical measure will be the following:

	$y = 0$	$y = 1$
$x = 1$	$1/4$	$1/4$
$x = 2$	$2/4$	0

while $p(y|x)$ will be following:

	$y = 0$	$y = 1$
$x = 1$	$1/2$	$1/2$
$x = 2$	1	0

4.3 Generative models

Types of generative models are:

- Naive Bayes classifier or Bayesian network
- Linear discriminant analysis

4.4 Discriminative models

- Logistic regression

- Support Vector Machines
- Decision Tree Learning
- Random Forest

5. Elaborate in detail about Support Vector Machine (SVM).

Illustrate the mathematical formulation for soft margin SVM hard margin SVM. [Nov 2024]

SUPPORT VECTOR MACHINE

5.1 Support Vector Machine (SVM)

5.2 Cost Function and Gradient Updates

5.2.1 Hinge loss function

5.3 SVM Kernel

5.4 Types of SVMs

5.5 Advantages of SVM

5.6 Disadvantages

5.7 Applications

5.1 Support Vector Machine (SVM)

- Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression.
- The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points.
- Hyperplanes are decision boundaries that help classify the data points.
- The dimension of the hyperplane depends upon the number of features.
- If the number of input features is 2, then the hyperplane is just a line.
- If the number of input features is 3, then the hyperplane becomes a two-dimensional plane.
- It becomes difficult to imagine when the number of features exceeds 3.

- The objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes.
- Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane.
- Using these support vectors, can maximize the margin of the classifier.
- Deleting the support vectors will change the position of the hyperplane.
- Example Refer Figure 2.10

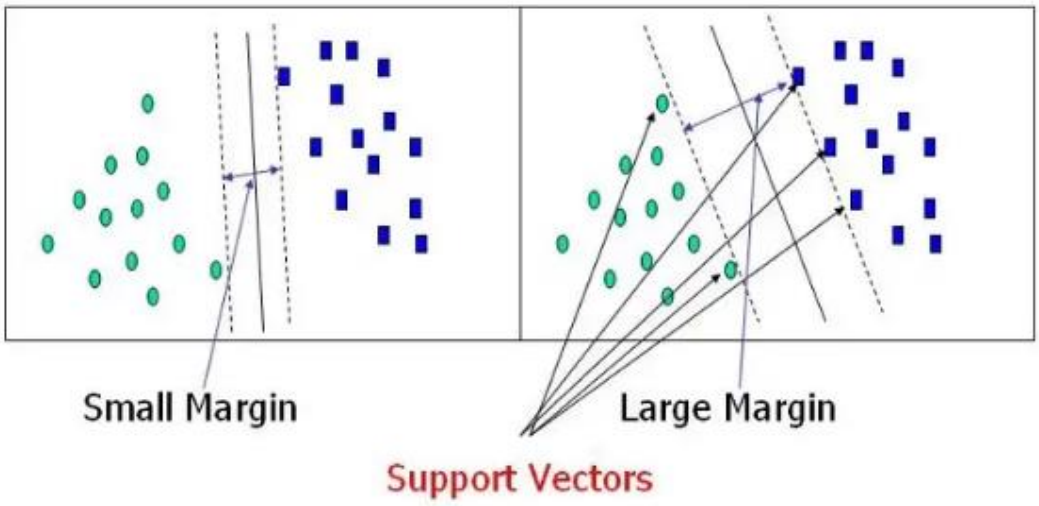


Figure 2.10 – Example for Support Vectors

- Let's consider two independent variables x1, x2 and one dependent variable which is either a blue circle or a red box as shown in figure 2.11.

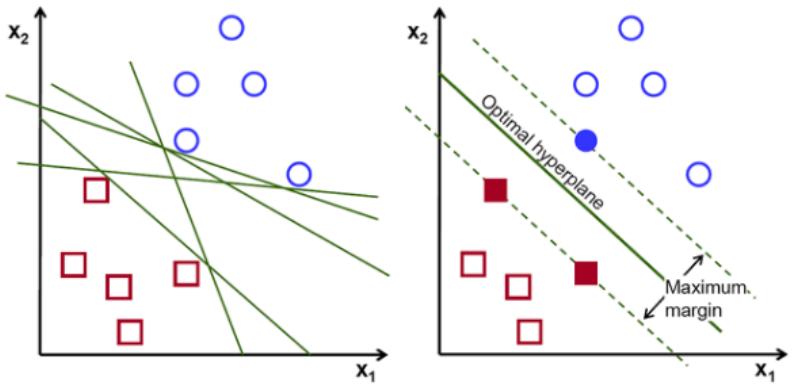


Figure 2.11 - Linearly Separable Data points

5.2 Cost Function and Gradient Updates

- In the SVM algorithm, to maximize the margin between the data points and the hyperplane, the loss function helps to maximize the margin is called hinge loss.

5.2.1 Hinge loss function

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

$$c(x, y, f(x)) = (1 - y * f(x))_+$$

- The cost is 0 if the predicted value and the actual value are of the same sign. If they are not, then calculate the loss value.
- The objective of the regularization parameter is to balance the margin maximization and loss.
- After adding the regularization parameter, the cost functions looks as below.

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

5.3 SVM Kernel:

- The SVM kernel is a function that takes low dimensional input space and transforms it into higher-dimensional space, ie it converts non separable problem to separable problem. It is mostly useful in non-linear separation problems.

5.4 Types of SVMs

- There are two different types of SVMs, each used for different things:
  - **Simple SVM:** Typically used for linear regression and classification problems.
  - **Kernel SVM**: Has more flexibility for non-linear data.

5.5 Advantages of SVM:

- Effective on datasets with multiple features, like financial or medical data.
- Effective in cases where number of features is greater than the number of data points.
- Its memory efficient as it uses a subset of training points in the decision function called support vectors
- Different kernel functions can be specified for the decision functions and its possible to specify custom kernels

5.6 Disadvantages

- If the number of features is a lot bigger than the number of data points, choosing kernel functions and regularization term is crucial.
- SVMs don't directly provide probability estimates. Those are calculated using an expensive five-fold cross-validation.
- Works best on small sample sets because of its high training time.

5.7 Applications

SVMs can be used to solve various real-world problems:

- SVMs are helpful in text and hypertext categorization.
- Classification of images can also be performed using SVMs.
- Classification of satellite data like SAR data using supervised SVM.
- Hand-written characters can be recognized using SVM.
- The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90% of the compounds classified correctly.

6. Elaborate in detail about Decision Tree in Supervised Learning.

With an example explain the Decision Tree Concepts in detail.

[Apr 2024]

Decision Tree in Supervised Learning

6.1 Decision Tree

6.1 Decision Tree

- Decision Tree is a supervised learning technique that can be used for both classification and Regression problems.
- It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, the Decision Node and Leaf Node.
- As shown in figure 2.12, Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- Example for Decision Tree Refer Figure 2.13

- The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data (training data).
- In order to build a tree, use the CART algorithm, which stands for Classification and Regression Tree algorithm.

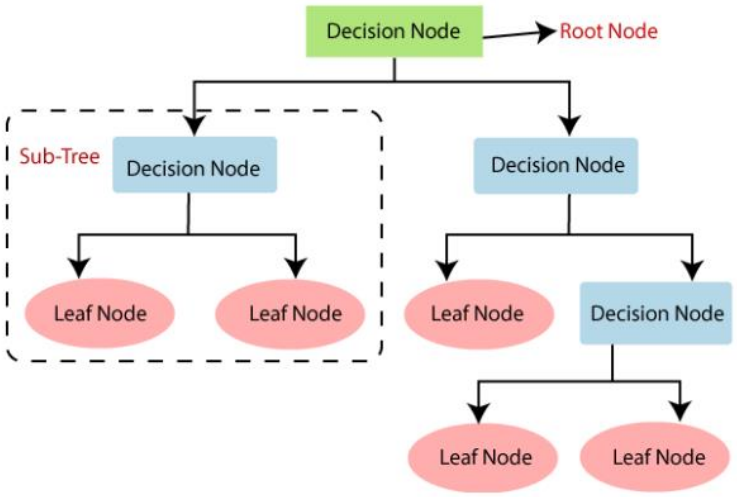


Figure 2.12 – Decision Tree Structure

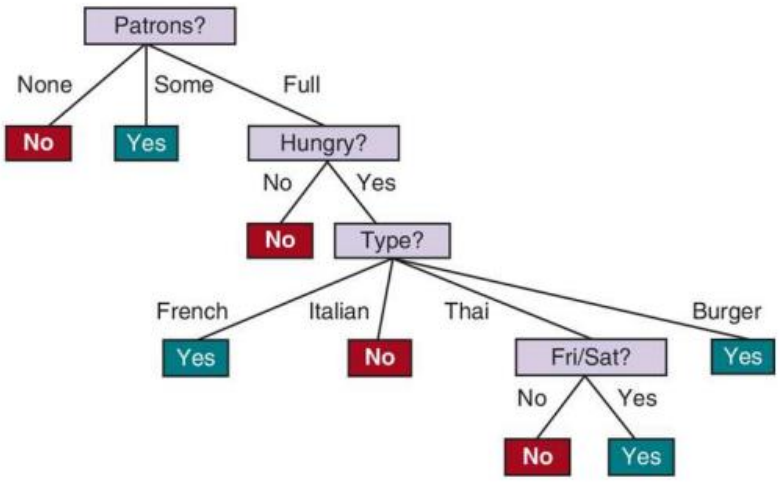


Figure 2.13 – Decision Tree Example

6.2 Types of Decision Trees

1. **Categorical Variable Decision Tree:** Decision Tree which has a categorical target variable then it called a **Categorical variable decision tree.**

2. **Continuous Variable Decision Tree:** Decision Tree has a continuous target variable then it is called **Continuous Variable Decision Tree.**

6.3 Reason for using Decision Trees

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

6.4 Decision Tree Terminologies

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

6.5 Working of Decision Tree algorithm

- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree.
- This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.
- For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further.
- It continues the process until it reaches the leaf node of the tree.
- The complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where cannot further classify the nodes and called the final node as a leaf node.

Example:

- Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM).
- The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels.
- The next decision node further gets split into one decision node (Cab facility) and one leaf node.
- Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram: Refer fig 2.14

Decision Tree

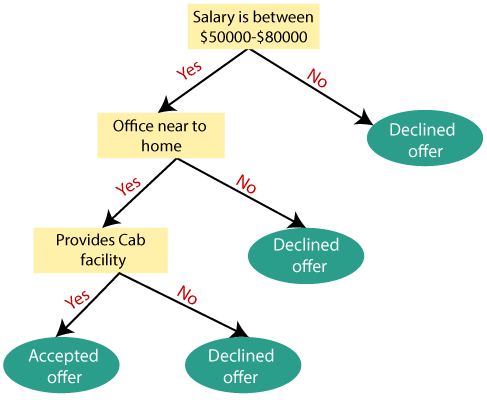


Figure 2.14 – Decision Tree Algorithm Example

6.6 Algorithms used to construct Decision Trees:

- ID3 → (extension of D3)
- C4.5 → (successor of ID3)
- CART → (Classification And Regression Tree)
- CHAID → (Chi-square automatic interaction detection Performs multi-level splits when computing classification trees)
- MARS → (multivariate adaptive regression splines)

6.7 Attribute Selection Measures

- While implementing a Decision tree, Attribute selection measure orASM is used to select the best attribute for the nodes of the tree.

1. Entropy,
2. Information gain,
3. Gini index,
4. Gain Ratio,
5. Reduction in Variance
6. Chi-Square

6.7.1 Entropy:

- Entropy is a metric to measure the impurity in a given attribute.
- Entropy is a measure of the randomness in the information being processed.
- The higher the entropy, the harder it is to draw any conclusions from that information.
- Flipping a coin is an example of an action that provides information that is random.
- Entropy can be calculated as:

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

Entropy(S) = -P(yes)log$_2$ P(yes)- P(no) log$_2$ P(no)

Where,

- S= Total number of samples
- P(yes)= probability of yes
- P(no)= probability of no

6.7.2. Information Gain:

- Information gain or IG is a statistical property that measures how well a given attribute separates the training examples according to their target classification. Example Refer Fig 3.15.

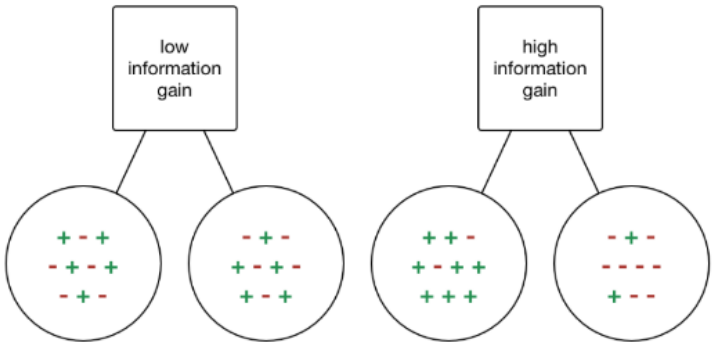


Figure 2.15 – Information Gain Example

- Constructing a decision tree is all about finding an attribute that returns the highest information gain and the smallest entropy.
- Information gain is a decrease in entropy.
- It computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values.
- It can be calculated using the below formula:

$$Information\ Gain\ =\ Entropy(before) - \sum_{j=1}^{K} Entropy(j,\ after)$$

Information Gain= Entropy(S)-[(Weighted Avg) *Entropy (each feature)]

6.7.3. Gini Index:

- Gini index as a cost function used to evaluate splits in the dataset.

- It is calculated by subtracting the sum of the squared probabilities of each class from one.
- It favors larger partitions and easy to implement whereas information gain favors smaller partitions with distinct values.
- Gini index can be calculated using the below formula:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

6.7.4 Gain Ratio

- Information gain is biased towards choosing attributes with a large number of values as root nodes.
- Gain ratio overcomes the problem with information gain by taking the intrinsic information of a split into account.

$$Gain\ Ratio = \frac{Information\ Gain}{SplitInfo} = \frac{Entropy\ (before) - \sum_{j=1}^{K} Entropy(j, after)}{\sum_{j=1}^{K} w_j \log_2 w_j}$$

6.7.5 Reduction in variance

- Reduction in variance is an algorithm that uses the standard formula of variance to choose the best split.
- The split with lower variance is selected as the criteria to split the population:

$$\text{Variance} = \frac{\Sigma(X - \overline{X})^2}{n}$$

6.7.6 Chi-Square

- The acronym CHAID stands for Chi-squared Automatic Interaction Detector.
- It finds out the statistical significance between the differences between sub-nodes and parent node.
- Higher the value of Chi-Square higher the statistical significance of differences between sub-node and Parent node.

- It generates a tree called CHAID (Chi-square Automatic Interaction Detector).
- Mathematically, Chi-squared is represented as:

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

Where:

χ^2 = Chi Square obtained
$\sum$ = the sum of
O = observed score
E = expected score

6.8. Avoid/counter Overfitting in Decision Trees

- Two ways to remove overfitting:
  - Pruning Decision Trees.
  - Random Forest

6.8.1 Pruning Decision Trees

- Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.
- A too-large tree increases the risk of over fitting, and a small tree may not capture all the important features of the dataset.
- Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning.
- There are mainly two types of tree pruning technology used:
  - Cost Complexity Pruning
  - Reduced Error Pruning.

6.8.2 Random Forest

- Random Forest is an example of ensemble learning, in which we combine multiple machine learning algorithms to obtain better predictive performance.
- The name random means

- A random sampling of training data set when building trees.
- Random subsets of features considered when splitting nodes.

6.9 Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

6.10 Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an over fitting issue, which can be resolved using the **Random Forest algorithm.**
- For more class labels, the computational complexity of the decision tree may increase.

7. Elaborate in detail about Random Forest in Supervised Learning.

Random Forest

7.1 Random Forest

7.2 Steps in the working process of Random Forest

7.3 Need for Random Forest

7.4 Example:

7.5 Important Features of Random Forest

7.6 Applications of Random Forest

7.8 Advantages of Random Forest

7.9 Disadvantages of Random Forest

7.10 Difference between Decision Tree & Random Forest

7.1 Random Forest

- Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.
- Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique.
- It can be used for both Classification and Regression problems in ML.
- It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
- The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting.

7.2 Steps in the working process of Random Forest

- The Working process can be explained in the below steps and diagram:

  Step 1: In Random forest n number of random records are taken from the data set having k number of records.

  Step 2: Individual decision trees are constructed for each sample.

  Step 3: Each decision tree will generate an output.

  Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

7.3 Need for Random Forest

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

7.4 Example

- Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier.
- The dataset is divided into subsets and given to each decision tree.

- During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision.

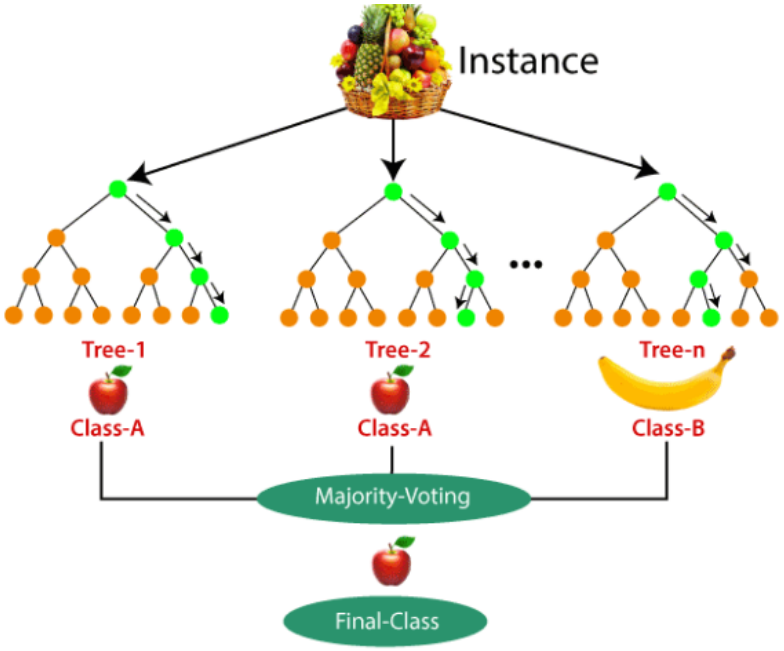


Figure 2.16 – Example for Random Forest

- In the above figure 2.16, the majority decision tree gives output as an apple when compared to a banana, so the final output is taken as an apple.

7.5 Important Features of Random Forest

1. Diversity

Not all attributes/variables/features are considered while making an individual tree, each tree is different.

2. Immune to the curse of dimensionality

Since each tree does not consider all the features, the feature space is reduced.

3. Parallelization

Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.

4. **Train-Test split**

   In a random forest we don't have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.

5. **Stability**

   Stability arises because the result is based on majority voting/ averaging.

7.6 Applications of Random Forest

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

7.7 Advantages of Random Forest

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the over fitting issue.

7.8 Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

7.9 Difference between Decision Tree & Random Forest

Decision trees	Random Forest

Decision trees normally suffer from the problem of overfitting if it's allowed to grow without any control.	Random forests are created from subsets of data and the final output is based on average or majority ranking and hence the problem of overfitting is taken care of.
A single decision tree is faster in computation.	It is comparatively slower.
When a data set with features is taken as input by a decision tree it will formulate some set of rules to do prediction.	Random forest randomly selects observations, builds a decision tree and the average result is taken. It doesn't use any set of formulas.

8. Compare Classification and Regression.

Feature	Classification	Regression
Output type	In this problem statement, the target variables are discrete. Discrete categories (e.g., "spam" or "not spam")	Continuous numerical value (e.g., price, temperature).
Goal	To predict which category a data point belongs to.	To predict an exact numerical value based on input data.
Example problems	Email spam detection, image recognition, customer sentiment analysis.	House price prediction, stock market forecasting, sales prediction.
Evaluation metrics	Evaluation metrics like Precision, Recall, and F1-Score	Mean Squared Error, R2-Score, , MAPE and RMSE.
Decision boundary	Clearly defined boundaries between different classes.	No distinct boundaries, focuses on finding the best fit line.
Common algorithms	Logistic regression, Decision trees, Support Vector Machines (SVM)	Linear Regression, Polynomial Regression, Decision Trees (with regression objective).

9. By the method of least squares find the straight line to the data given below. [Apr 2024]

X	5	10	15	20	25
Y	16	19	23	26	30

Let us consider, the straight line is, $y = ax + b$.

The normal equations are:

$a \sum x + 5b = \sum y$ .....(1)

$a \sum x\ 2 + b \sum x = \sum xy$ .....(2)

Now, we have to calculate $\sum x$, $\sum y$, $\sum x\ 2$, $\sum xy$ and so we form the following table

	x	y	x^2	xy
	5	16	25	80
	10	19	100	190
	15	23	225	345
	20	26	400	520
	25	30	625	750
Total	75	114	1375	1885

The normal equations are :

75a + 5b = 114 .....(3)

1375a + 75b = 1885 .....(4)

Solving (3) and (4), we get, a = 0.7, b = 12.3

Hence, the best fitting line is y = 0.7x + 12.3 ..... (P)

By Second Form :

Let us consider,

$X = \frac{x-a}{h} = \frac{x-15}{5}$ where, $a = 15$ (middle point of the column x)

$Y = \frac{y-b}{h} = \frac{y-23}{5}$ where, $b = 23$ (middle point of the column y)

Let us take, the line in new variable is : $Y = AX$ + B

	x	y	X	X^2	Y	XY
	5	16	−2	4	−1.4	2.8
	10	19	−1	1	−0.8	0.8
	15	23	0	0	0	0
	20	26	1	1	0.6	0.6
	25	30	2	4	1.4	2.8
Total			0	10	−0.2	7

The normal equations are :

$$A\sum X + 5B = \sum Y \quad \text{.....(5)}$$

$$A\sum X^2 + B\sum X = \sum XY \quad \text{.....(6)}$$

Solving (5) and (6), we get, A = 0.7 and B = −0.04

∴ $Y = 0.7X - 0.04$

$$\Rightarrow \quad \frac{y-23}{5} = 0.7 \times \frac{x-15}{5} - 0.04 = y - 23 = 0.7x - 10.5 - 0.2$$

$$\Rightarrow \quad y = 0.7x + 33.3 \quad \text{.....(Q)}$$

Thus, the equations (P) and (Q) are same.

10. Discuss the model representation for logistic regression. [Nov 2024]

Model Representation for Logistic Regression

- Logistic Regression is a statistical model used for binary classification, where the output is either 0 or 1.
- Unlike linear regression, which predicts continuous values, logistic regression predicts the probability that a given input belongs to a particular class.

The model representation consists of the following components:

1. Hypothesis Function

- Logistic regression models the probability of a class label using the sigmoid function (logistic function):

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

where:

- $h_\theta(x)$ is the predicted probability that $y = 1$ given x.
- θ is the parameter vector (weights).
- x is the feature vector (including $x_0 = 1$ for bias).
- e is Euler's number (approximately 2.718).

- The sigmoid function ensures that the output is between 0 and 1, making it suitable for probability estimation.

2. Decision Boundary

- The model makes a classification decision by setting a threshold (typically **0.5**).
- If $h_\theta(x) \geq 0.5$, predict **1** (positive class).
- If $h_\theta(x) < 0.5$, predict **0** (negative class).

Mathematically, the decision rule is:

$$y = \begin{cases} 1, & \text{if } h_\theta(x) \geq 0.5 \\ 0, & \text{otherwise} \end{cases}$$

Since $h_\theta(x)$ is a function of $\theta^T x$, the decision boundary is defined by the equation:

$$\theta^T x \downarrow 0$$

This represents a linear decision boundary in the feature space.

3. Cost Function

Instead of using Mean Squared Error (MSE), logistic regression uses the log loss (cross-entropy loss):

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

where:

- m is the number of training examples.
- $y^{(i)}$ is the actual label (0 or 1).
- $h_\theta(x^{(i)})$ is the predicted probability.

This cost function is convex, ensuring that gradient descent can find the global minimum efficiently.

4. Optimization (Gradient Descent)

To minimize the cost function, gradient descent is used to update the model parameters:

$$\theta_j := \theta_j - \alpha\frac{\partial J(\theta)}{\partial \theta_j}$$

for each parameter θ_j, where:

- α is the learning rate.
- The partial derivative of the cost function gives the gradient:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

5. Interpretation of Model Parameters

- Each coefficient θ_j represents the **log-odds** of the outcome changing when the corresponding feature x_j changes.
- The odds ratio is given by:

$$e^{\theta_j}$$

which tells how the odds of $y = 1$ change with a unit increase in x_j.

- Logistic regression is a simple yet powerful classification model that maps inputs to probabilities using the sigmoid function. Its decision boundary is linear, and it is trained using gradient descent on a convex log-loss function.

12. Derive the gradient of cost function (error) for logistic regression which uses binary cross entropy. [Nov 2024]

- Gradient Derivation of Cost Function for Logistic Regression (Binary Cross-Entropy Loss)
- In logistic regression, we use the binary cross-entropy loss function to measure the difference between predicted probabilities and actual labels. Our goal is to derive the gradient of this cost function with respect to the model parameters θ
- θ, which will be used in gradient descent for optimization.

Step 1: Define the Cost Function

- The cost function for logistic regression is the binary cross-entropy (log loss):

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

where:

- m is the number of training examples.
- $y^{(i)}$ is the actual label (either **0** or **1**).
- $h_\theta(x^{(i)})$ is the predicted probability, given by ↓ **sigmoid function**:

$$h_\theta(x^{(i)}) = \frac{1}{1 + e^{-\theta^T x^{(i)}}}$$

- $x^{(i)}$ is the feature vector of the i-th example.
- θ is the parameter vector.

Our objective is to compute:

$$\frac{\partial J(\theta)}{\partial \theta_j}$$

for each parameter θ_j.

Step 2: Compute the Partial Derivative of Cost Function

To derive the gradient, we differentiate $J(\theta)$ with respect to θ_j:

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial \theta_j} \left[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

Using the chain rule, we differentiate each term separately.

Step 2.1: Differentiate the First Term

Since:

$$h_\theta(x^{(i)}) = \frac{1}{1 + e^{-\theta^T x^{(i)}}}$$

its derivative with respect to θ_j is:

$$\frac{\partial h_\theta(x^{(i)})}{\partial \theta_j} = h_\theta(x^{(i)})(1 - h_\theta(x^{(i)}))x_j^{(i)}$$

Now, differentiate:

$$\frac{\partial}{\partial \theta_j} \left[y^{(i)} \log h_\theta(x^{(i)}) \right]$$

Using the chain rule:

$$y^{(i)} \cdot \frac{1}{h_\theta(x^{(i)})} \cdot \frac{\partial h_\theta(x^{(i)})}{\partial \theta_j}$$

Substituting:

$$y^{(i)} \cdot \frac{1}{h_\theta(x^{(i)})} \cdot h_\theta(x^{(i)})(1 - h_\theta(x^{(i)}))x_j^{(i)}$$

Simplifies to:

$$y^{(i)}(1 - h_\theta(x^{(i)}))x_j^{(i)}$$

Step 2.2: Differentiate the Second Term

$$\frac{\partial}{\partial \theta_j}\left[(1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right]$$

Using the chain rule:

$$(1 - y^{(i)}) \cdot \frac{1}{1 - h_\theta(x^{(i)})} \cdot \left(-\frac{\partial h_\theta(x^{(i)})}{\partial \theta_j}\right)$$

Substituting:

$$(1 - y^{(i)}) \cdot \frac{1}{1 - h_\theta(x^{(i)})} \cdot (-h_\theta(x^{(i)})(1 - h_\theta(x^{(i)}))x_j^{(i)})$$

Simplifies to:

$$-(1 - y^{(i)})h_\theta(x^{(i)})x_j^{(i)}$$

Step 3: Combine Both Terms

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}(1 - h_\theta(x^{(i)}))x_j^{(i)} - (1 - y^{(i)})h_\theta(x^{(i)})x_j^{(i)}\right]$$

Factor out $x_j^{(i)}$:

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}\left[y^{(i)}(1 - h_\theta(x^{(i)})) - (1 - y^{(i)})h_\theta(x^{(i)})\right]$$

Rewriting inside the brackets:

$$y^{(i)} - y^{(i)}h_\theta(x^{(i)}) - h_\theta(x^{(i)}) + y^{(i)}h_\theta(x^{(i)})$$

$$y^{(i)} - h_\theta(x^{(i)})$$

Thus, the gradient simplifies to:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Step 4: Vectorized Form

Using matrix notation, define:

- X as the $m \times n$ feature matrix.
- y as the $m \times 1$ label vector.
- h_θ as the $m \times 1$ vector of predictions.

Then, the gradient can be written as:

$$\nabla J(\theta) = \frac{1}{m} X^T (h_\theta - y)$$

where:

- X^T is the transpose of the feature matrix.
- $h_\theta - y$ is the vector of prediction errors.

MAILAM Engineering College

Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai,

Accredited by National Board of Accreditation (NBA), Accredited by NAAC with "A" Grade &

Accredited by TATA Consultancy Services (TCS), Chennai)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

II YEAR / IV SEM

AL3451- MACHINE LEARNING

UNIT III ENSEMBLE TECHNIQUES AND UNSUPERVISED LEARNING

SYLLABUS:

Combining multiple learners: Model combination schemes, Voting, Ensemble Learning - bagging, boosting, stacking, Unsupervised learning: K-means, Instance Based Learning: KNN, Gaussian mixturemodels and Expectation maximization

PART A

1. When is supervised learning better than unsupervised learning? [Apr 2024]

- Supervised learning is better than unsupervised learning when you have labeled data and a clear goal to predict a specific outcome, such as classifying emails as spam or not spam, identifying objects in an image, or forecasting future values based on historical data;
- Essentially, when you need the model to learn from known examples to make accurate predictions on new data.

2. Define Expectation Maximization. [Apr 2024]

- Expectation maximization (EM) is an iterative algorithm that estimates parameters in statistical models with missing or hidden data.
- It's a type of maximum likelihood estimation.

- In statistics, an expectation–maximization (EM) algorithm is an iterative method to find (local) maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables.

3. Write the purpose of bagging in ensemble learning. [Nov 2024]

- Bagging, also known as bootstrap aggregation, is the ensemble learning method that is commonly used to reduce variance within a noisy data set. In bagging, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once.

4. Define unsupervised learning. [Nov 2024]

- Unsupervised learning is a machine learning technique that uses unlabeled data to discover patterns and relationships without human intervention.
- Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses.

5. What is unsupervised learning?

- Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses.

6. What is semi supervised learning?

- Semi-supervised machine learning is **a combination of supervised and unsupervised learning**.
- It uses a small amount of labeled data and a large amount of unlabeled data, which provides the benefits of both unsupervised and supervised learning while avoiding the challenges of finding a large amount of labeled data.

7. What is Ensemble method?

- Ensemble methods are techniques that aim at ***improving the accuracy of results in models by combining multiple models instead of using a single model***. The combined models increase the accuracy of the results significantly. This has boosted the popularity of ensemble methods in machine learning.

8. Explain Clustering.

- Clustering is **the act of organizing similar objects into groups within a machine learning algorithm**. Assigning related objects into clusters is beneficial for AI models. Clustering has many uses in data science, like Image

processing, knowledge discovery in data, unsupervised learning, and various other applications.

9. What is Cluster?

- Cluster is a group of objects that belongs to the same class. In other words the similar objects are grouped in one cluster and dissimilar are grouped in other cluster.

10. What is bagging?

- Bagging is also known as Bootstrap aggregation, ensemble method works by training multiple models independently and combining later to result in strong model.

11. Define Boosting.

- Boosting refers to a group of algorithms that utilize weighted averages to make weak learning algorithms to stronger learning algorithms.

12. What is K-Nearest neighbor Methods?

- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

13. Which are the performance factors that influence KNN algorithm?

1. The distance function or distance metric used to determine the nearest neighbors

2. The Decision rule used to derive a classification from the K-Nearest neighbors.

3. The number of neighbors used to classify the new example.

14. What is K Means Clustering?

- K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

- It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

15. List the properties of K-Means algorithm.

1. There are always K clusters
2. There is always at least one item in each cluster.
3. The clusters are non-hierarchical and they do not overlap

16. What is stacking?

- Stacking, sometimes called stacked generalization, is an ensemble machine learning method that combines heterogeneous base or component models via a meta model.

17. How do GMMs differentiate from K-means clustering?

- GMMs and K-means, both are clustering algorithms used for unsupervised learning tasks. However, the basic difference between them is that K-means is a distance-based clustering method while GMMs is a distribution based clustering method.

18. What is 'Over fitting' in Machine learning?

- In machine learning, when a statistical model describes random error or noise instead of underlying relationship 'over fitting' occurs.
- When a model is excessively complex, over fitting is normally observed, because of having too many parameters with respect to the number of training data types. The model exhibits poor performance which has been over fit.

19. What is ensemble learning?

- To solve a particular computational program, multiple models such as classifiers or experts are strategically generated and combined. This process is known as ensemble learning.

20. What are the two paradigms of ensemble methods?

The two paradigms of ensemble methods are

- Sequential ensemble methods
- Parallel ensemble methods

21. What is voting?

- A voting classifier is a machine learning estimator that trains various base models or estimators and predicts on the basis of aggregating the findings of each base estimator. The aggregating criteria can be combined decision of voting for each estimator output.

22. What is Error-Correcting Output Codes?

- The main classification task is defined in terms of a number of subtasks that are implemented by the base learners.
- The idea is that the original task of separating oneclass from all other classes may be a difficult problem.
- We want to define a set of simpler classification problems, each specializing in one aspect of the task, and combining these simpler classifiers, we get the final classifier.

$$y_i = \sum_{j=1}^{L} w_{ij} d_j$$

23. What is Gaussian Mixture models?

- This model is a soft probabilistic clustering model that allows us to describe the membership of points to a set of clusters using a mixture of Gaussian densities.

24. Differentiate between Bagging and Boosting.

Sl no	Bagging	Boosting
1.	The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
2.	Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
3.	Each model receives equal weight.	Models are weighted according to their performance.
4.	Each model is built independently.	New models are influenced by the performance of previously built

PART - B

1. Give Short notes on combining multiple learners.

- Combining multiple learners is a machine learning technique called ensemble learning. The goal is to improve the performance of a model by combining multiple machine learning models into a single model.

Rationale

✓ The **No Free Lunch Theorem** states that there is no single learning algorithm that in any domain always induces the most accurate learner. The usual approach is to try many and choose the one that performs the best on a separate validation set.

Different Learners use Different

- **Algorithms:** making different assumptions
- **Hyper parameters:** e.g number of hidden nodes in NN, k in k-NN
- **Representations:** different features, multiple sources of information
- **Training sets:** small variations in the sets or different subproblems

Different Algorithms

o Different algorithms make different assumptions about the data and lead to different classifiers.

Different Hyper parameters

o Use the same learning algorithm but use it with different hyper parameters.

Different Input Representations

o Separate base-learners may be using different *representations* of the same input object or event, making it possible to integrate different types of Sensors/measurements/modalities.

o Different representations make different characteristics explicit allowing better identification.

Different Training Sets

o Another possibility is to train different base-learners by different subsets of the training set. This can be done randomly by drawing random training sets from the given sample this is called *bagging*.

Model Combination Schemes

- ✓ There are also different ways the multiple base-learners are combined to generate the final output:

➢ **Multiexpert combination**

- Multiexpert combination methods have base-learners that work in parallel. methods can in turn be divided into two:

A) The global approach, also called learner fusion, given an input, all base-learners generate an output and all these outputs are used. Examples are voting and stacking.

B) The local approach, or learner selection, for example, in mixture of experts, there is a gating model, which looks at the input and chooses one (or very few) of the learners as responsible for generating the output.

➢ **Multistage combination methods** use a serial approach where the next multistage combination base-learner is trained with or tested on only the instances where the previous base-learners are not accurate enough.

- Let *L be the* base-learners, $dj(x)$ the prediction of base-learner, *Mj* given the arbitrary dimensional input *x*
- Each *Mj* uses a different input representation *xj*.
- The final prediction is calculated from the predictions of the baselearners:

$$Y= f(d1,d2,\ldots,d_L|\Phi)$$

- where $f(\cdot)$ is the combining function with Φ denoting its parameters.
- When there are *K* outputs, for each learner there are $dji(x)$, $i= 1, \ldots, K$, $j = 1, \ldots, L$, and, combining them, also generate *K* values, *yi*, $i = 1, \ldots, K$ and then for example in classification, choose the class with the maximum *yi* value:

Voting

- A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output.

- It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting.
- The idea is instead of creating separate dedicated models and finding the accuracy for each them, create a single model which trains by these models and predicts output based on their combined majority of voting for each output class.
- The simplest way to combine multiple classifiers is by *voting*, which corresponds to taking a linear combination of the learners (see figure 4.1)
- This is also known as ensembles. In the simplest case, all learners are given equal weight and simple voting corresponds to take an average.

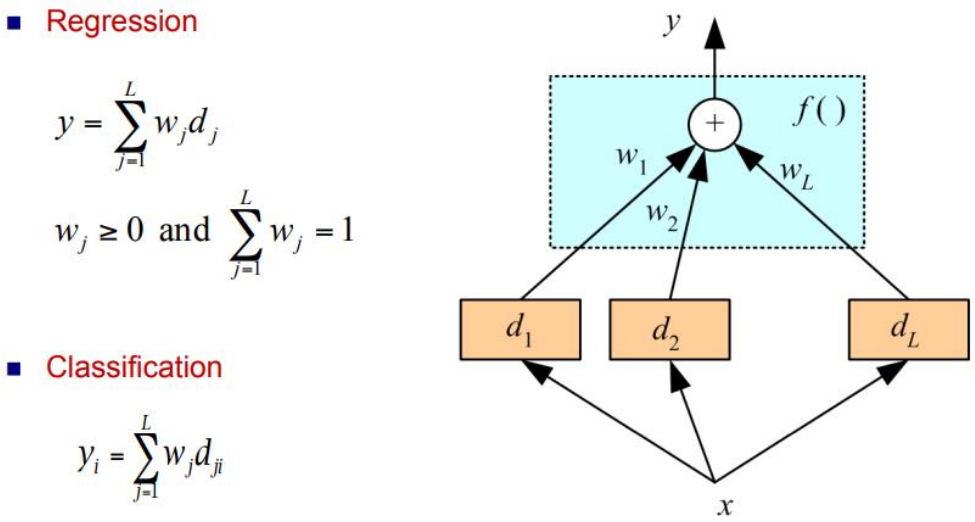


Figure 3.1 Base-learners are dj and their outputs are combined using f(·).

- Voting Classifier supports two types of votings.

1. **Hard Voting:** In hard voting, the predicted output class is a class with the highest majority of votes i.e the class which had the highest probability of being predicted by each of the classifiers. Suppose three classifiers predicted the *output class(A, A, B)*, so here the majority predicted *A* as output. Hence *A* will be the final prediction.
2. **Soft Voting:** In soft voting, the output class is the prediction based on the average of probability given to that class. Suppose given some input to three

models, the prediction probability for class *A = (0.30, 0.47, 0.53)* and *B = (0.20, 0.32, 0.40)*. So the average for class *A is 0.4333* and *B is 0.3067*, the winner is clearly class *A* because it had the highest probability averaged by each classifier.

Rule	Fusion function $f(\cdot)$
Sum	$y_i = \frac{1}{L}\sum_{j=1}^{L} d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji}, w_j \geq 0, \sum_j w_j = 1$
Median	$y_i = \text{median}_j\, d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$

Table 3.1 Classifier combination rules

- Voting schemes can be seen as approximations under a Bayesian framework with weights approximating prior model probabilities, and model decisions approximating model conditional likelihoods. This is *Bayesian model combination*

$$P(C_i|x) = \sum_{\text{all models } \mathcal{M}_j} P(C_i|x, \mathcal{M}_j) P(\mathcal{M}_j)$$

2. Explain Ensemble learning Technique in detail.

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

Simple Ensemble Training Methods

- Simple ensemble training methods typically just involve the application of statistical summary techniques, such as determining the mode, mean, or weighted average of a set of predictions.

Advanced Ensemble Training Methods

- **Bagging,** that often considers homogeneous weak learners, learns them independently from each other in parallel and combines them following some kind of deterministic averaging process
- **Boosting**, that often considers homogeneous weak learners, learns them sequentially in a very adaptive way (a base model depends on the previous ones) and combines them following a deterministic strategy
- **Stacking**, that often considers heterogeneous weak learners, learns them in parallel and combines them by training a meta-model to output a prediction based on the different weak models predictions

Ensemble learning methods can be divided into one of two different groups:

1. Sequential methods

- In Sequential ensemble methods the base learners/models are generated sequentially.
- In the case of sequential methods, the essential idea is that the dependence between the base learners is exploited in order to get more accurate predictions.
- Examples of sequential ensemble methods include AdaBoost, XGBoost, and Gradient tree boosting.

2. Parallel ensemble methods

- Parallel ensemble methods generate the base learners in parallel.
- When carrying out parallel ensemble learning, the idea is to exploit the fact that the base learners have independence, as the general error rate can be reduced by averaging the predictions of the individual learners.

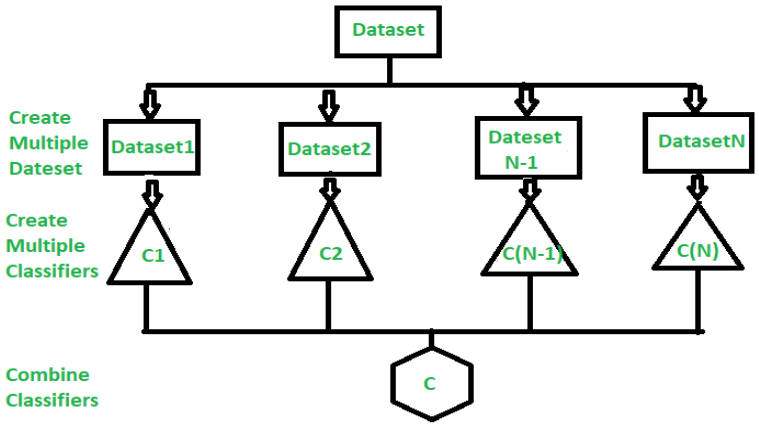


Fig.3.2 Parallel ensemble methods

Methods for Independently Constructing Ensembles –

- Majority Vote
- Bagging and Random Forest
- Randomness Injection
- Feature-Selection Ensembles
- Error-Correcting Output Coding

Methods for Coordinated Construction of Ensembles –

- Boosting
- Stacking

3. Explain in Detail about Bagging Technique in Ensemble Learning.

Bagging

- Bagging, also known as Bootstrap aggregating, is the aggregation of multiple versions of a predicted model and an ensemble learning technique that helps to improve the performance and accuracy of machine learning algorithms.
- Each model is trained individually, and combined using an averaging process.
- The primary focus of bagging is to achieve less variance than any model has individually.
- Bagging avoids over fitting of data and is used for both regression and classification models, specifically for decision tree algorithms.

Bootstrapping

- Bootstrapping is the process of generating bootstrapped samples from the given dataset.
- The samples are formulated by randomly drawing the data points with replacement.

Fig.3.3 Bootstraping

Implementation Steps of Bagging

- Step 1: Multiple subsets are created from the original data set with equal tuples, selecting observations with replacement.
- Step 2: A base model is created on each of these subsets.
- Step 3: Each model is learned in parallel with each training set and independent of each other.
- Step 4: The final predictions are determined by combining the predictions from all the models.

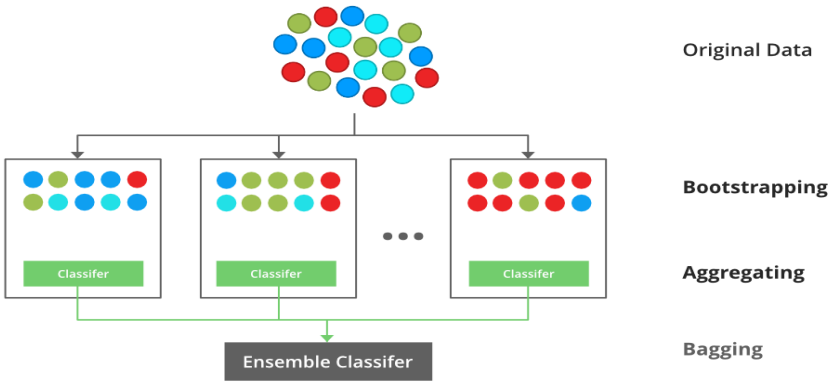


Figure 3.4 Bagging Technique.

Advantages:

1. Reduce overfitting of the model.
2. Handles higher dimensionality data very well.
3. Maintains accuracy for missing data

Disadvantage:

- Since final prediction is based on mean prediction from the subset trees, it won't give precise values for the classification and regression model

4. Explain boosting Technique in Ensemble learning.

- Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers.
- It is done by building a model by using weak models in series.
- Firstly, a model is built from the training data.
- Then the second model is built which tries to correct the errors present in the first model.
- This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models is added.
- AdaBoost is short for Adaptive Boosting and is a very popular boosting technique that combines multiple "weak classifiers" into a single "strong classifier".

Algorithm

1. *Initialize the dataset and assign equal weight to each of the data point.*
2. *Provide this as input to the model and identify the wrongly classified data points.*
3. *Increase the weight of the wrongly classified data points and decrease the weights of correctly classified data points. And then normalize the weights of all data points.*
4. *if (got required results)*

   Goto step 5

   else

   Goto step 2
5. *End*

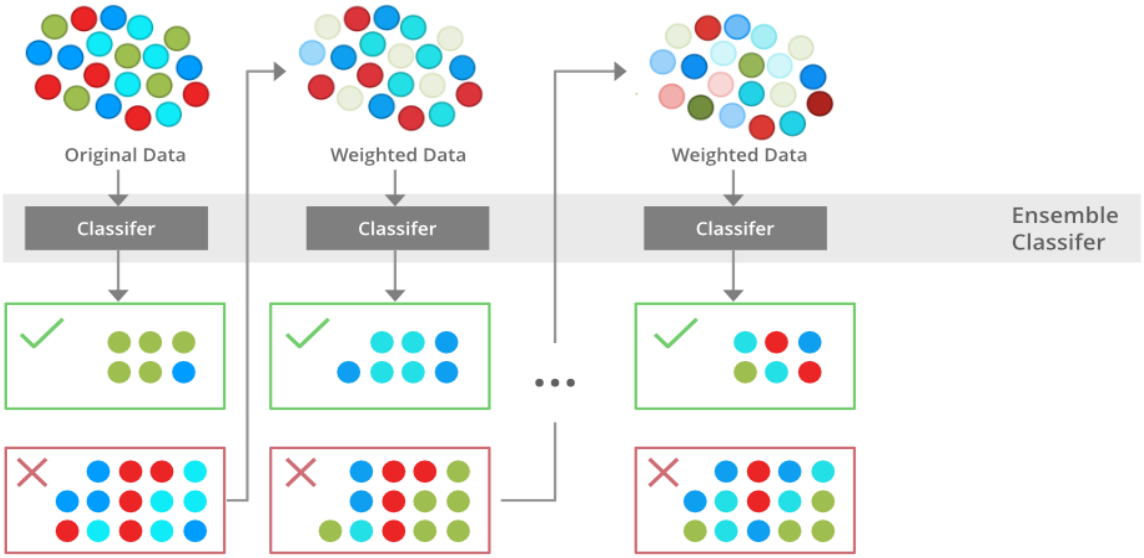


Figure 3.5 An illustration presenting the intuition behind the boosting algorithm, consisting of the parallel learners and weighted dataset.

Similarities Between Bagging and Boosting

1. Both are ensemble methods to get N learners from 1 learner.
2. Both generate several training data sets by random sampling.
3. Both make the final decision by averaging the N learners (or taking the majority of them i.e Majority Voting).
4. Both are good at reducing variance and provide higher stability.

Differences between Bagging and Boosting

Sl no	Bagging	Boosting
1.	The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.

Sl no	Bagging	Boosting
2.	Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
3.	Each model receives equal weight.	Models are weighted according to their performance.
4.	Each model is built independently.	New models are influenced by the performance of previously built models.
5	Example: The Random forest model uses Bagging.	Example: The AdaBoost uses Boosting techniques
6	In this base classifiers are trained parallelly.	In this base classifiers are trained sequentially.

5. Explain in Detail about Stacking.

- Stacking is one of the most popular ensemble machine learning techniques used to predict multiple nodes to build a new model and improve model performance.
- Various weak learners are ensemble in a parallel manner in such a way that by combining them with Meta learners, can predict better predictions for the future.
- Stacking enables us to train multiple models to solve similar problems, and based on their combined output, it builds a new model with improved performance.
- Stacking is a way of ensembling classification or regression models it consists of two-layer estimators.

- The first layer consists of all the baseline models that are used to predict the outputs on the test datasets.
- The second layer consists of Meta-Classifier or Regressor which takes all the predictions of baseline models as an input and generate new predictions.

Architecture of Stacking

- The architecture of the stacking model consists of two or more base/learner's models and a meta-model that combines the predictions of the base models.
- These base models are called level 0 models, and the meta-model is known as the level 1 model.
- So, the Stacking ensemble method includes **original (training) data, primary level models, primary level prediction, secondary level model, and final prediction**.
- The basic architecture of stacking can be represented as shown in fig.3.6

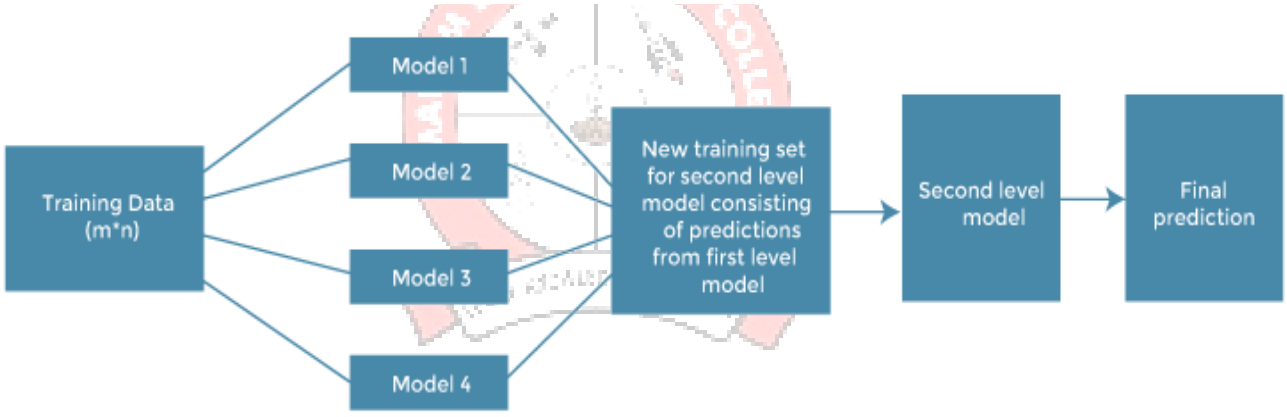


Fig.3.6 Basic architecture of Stacking

- **Original data:** This data is divided into n-folds and is also considered test data or training data.
- **Base models:** These models are also referred to as level-0 models. These models use training data and provide compiled predictions (level-0) as an output.
- **Level-0 Predictions:** Each base model is triggered on some training data and provides different predictions, which are known as **level-0 predictions.**
- **Meta Model:** The architecture of the stacking model consists of one meta-model, which helps to best combine the predictions of the base models. The meta-model is also known as the **level-1 model**.

- **Level-1 Prediction:** The meta-model learns how to best combine the predictions of the base models and is trained on different predictions made by individual base models, i.e., data not used to train the base models are fed to the meta-model, predictions are made, and these predictions, along with the expected outputs, provide the input and output pairs of the training dataset used to fit the meta-model.

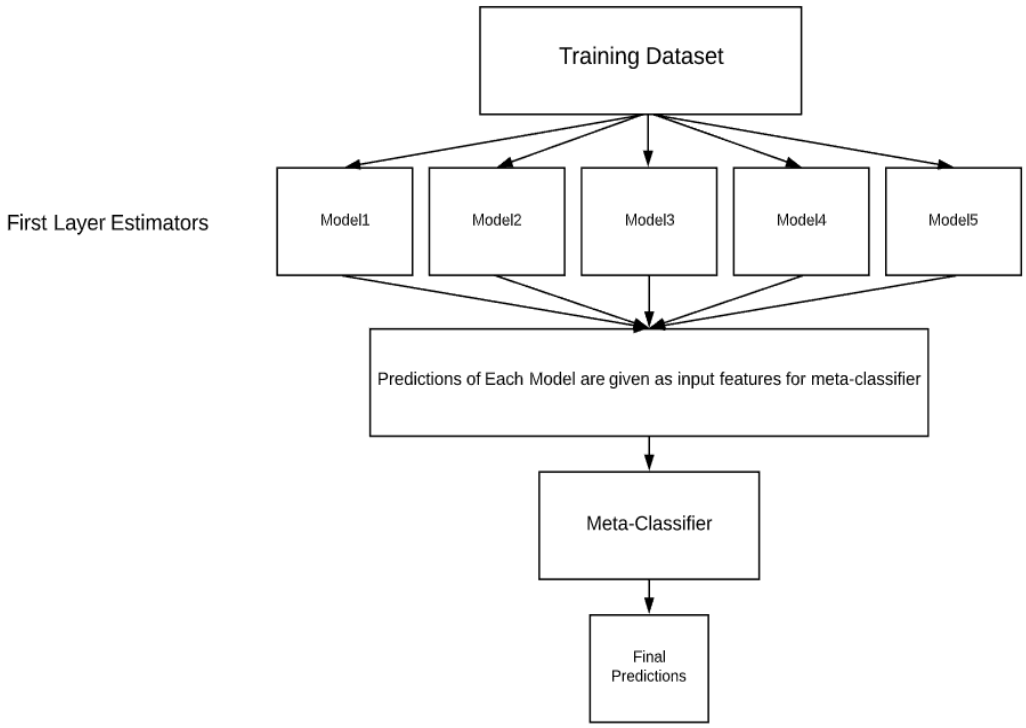


Figure 3.7 Stacking Architecture

Steps to implement Stacking models:

- Split training data sets into n-folds using the **Repeated Stratified KFold** as this is the most common approach to preparing training datasets for meta-models.
- Now the base model is fitted with the first fold, which is n-1, and it will make predictions for the nth folds.
- The prediction made in the above step is added to the x1_train list.
- Repeat steps 2 & 3 for remaining n-1folds, so it will give x1_train array of size n,
- Now, the model is trained on all the n parts, which will make predictions for the sample data.

- Add this prediction to the y1_test list.
- In the same way, we can find x2_train, y2_test, x3_train, and y3_test by using Model 2 and 3 for training, respectively, to get Level 2 predictions.
- Now train the Meta model on level 1 prediction, where these predictions will be used as features for the model (refer Figure 4.3).
- Finally, Meta learners can now be used to make a prediction on test data in the stacking model.

6. Explain in detail about Unsupervised Learning.

- Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision.

Clustering

- Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset.
- It is "A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."
- Clustering involves dividing data points into multiple clusters of similar values.
- Cluster is a group of objects that belongs to the same class
- Clustering is a process of partitioning a set of data in a meaningful subclass.

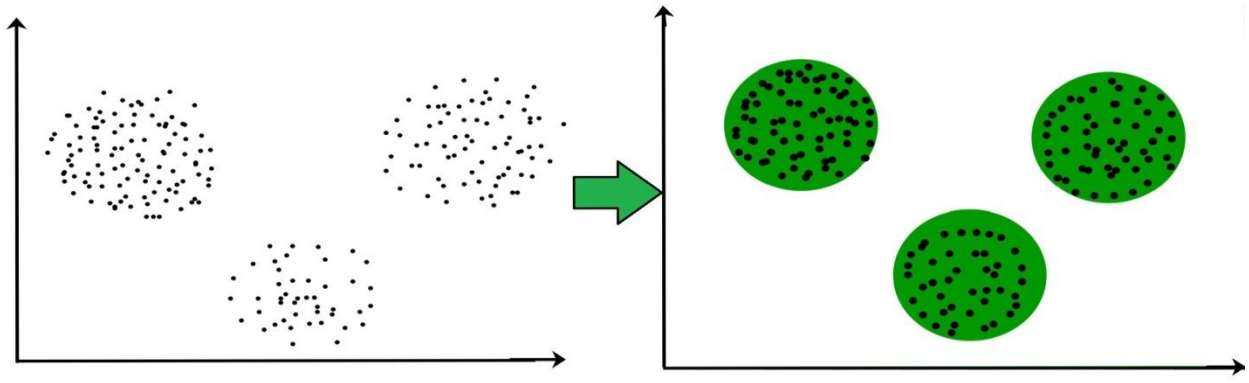


Figure 3.8 Clustering

Common uses of this technique are:

- Market Segmentation
- Statistical data analysis
- Social network analysis
- Image segmentation
- Anomaly detection, etc.

Types of Clustering Methods

- Hard clustering (datapoint belongs to only one group)
- Soft Clustering (data points can belong to another group also).

Main clustering methods used in Machine learning:

- Partitioning Clustering
- Density-Based Clustering
- Distribution Model-Based Clustering
- Hierarchical Clustering
- Fuzzy Clustering

Partitioning Clustering

- It is a type of clustering that divides the data into non-hierarchical groups.
- It is also known as the **centroid-based method**.
- The most common example of partitioning clustering is the **K-Means Clustering algorithm**.

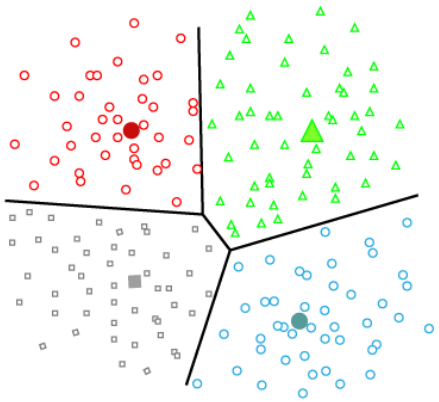


Fig.3.9 Clustering

Density-Based Clustering

- The density-based clustering method connects the highly-dense areas into clusters, and the arbitrarily shaped distributions are formed as long as the dense region can be connected.
- This algorithm does it by identifying different clusters in the dataset and connects the areas of high densities into clusters.
- The dense areas in data space are divided from each other by sparser areas.

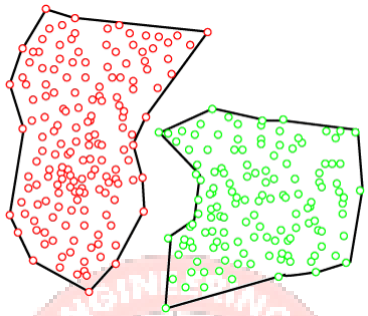


Fig.3.10 Density Based Clustering

Distribution Model-Based Clustering

- In the distribution model-based clustering method, the data is divided based on the probability of how a dataset belongs to a particular distribution.
- The grouping is done by assuming some distributions commonly **Gaussian Distribution**.
- The example of this type is the **Expectation-Maximization Clustering algorithm** that uses Gaussian Mixture Models (GMM).

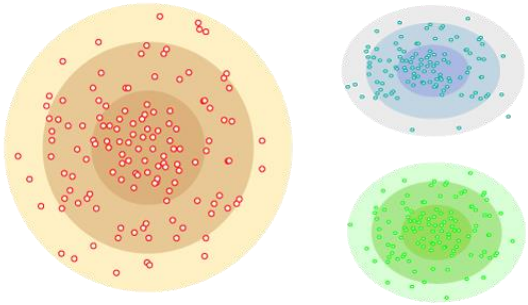


Fig.3.11 Distribution Model Based Clustering

Hierarchical Clustering

- In this technique, the dataset is divided into clusters to create a tree-like structure, which is also called a **dendrogram**.
- The observations or any number of clusters can be selected by cutting the tree at the correct level.
- The most common example of this method is the **Agglomerative Hierarchical algorithm**.

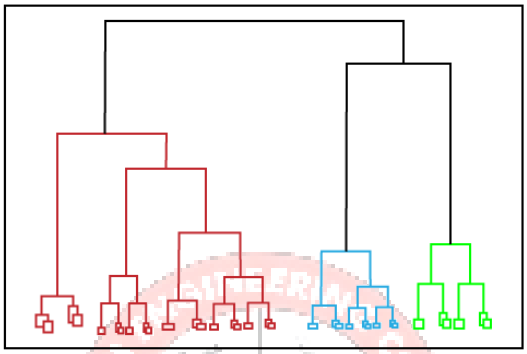


Fig.3.12 Hierarchical Clustering

Fuzzy Clustering

- Fuzzy clustering is a type of soft method in which a data object may belong to more than one group or cluster.
- Each dataset has a set of membership coefficients, which depend on the degree of membership to be in a cluster.
- **Fuzzy C-means algorithm** is the example of this type of clustering; it is sometimes also known as the Fuzzy k-means algorithm.

Clustering Algorithms

- K-Means algorithm
- Mean-shift algorithm
- DBSCAN Algorithm-Density-Based Spatial Clustering of Applications with Noise.
- Expectation-Maximization Clustering using GMM
- Agglomerative Hierarchical algorithm
- Affinity Propagation

Applications of Clustering

- In Identification of Cancer Cells: It divides the cancerous and non-cancerous data sets into different groups.
- In Search Engines: It does it by grouping similar data objects in one group that is far from the other dissimilar objects.
- Customer Segmentation: It is used in market research to segment the customers based on their choice and preferences.
- In Biology: It is used in the biology stream to classify different species of plants and animals using the image recognition technique.
- In Land Use: The clustering technique is used in identifying the area of similar lands use in the GIS database.

Unsupervised Learning : K means

- K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters.
- Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.
- It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.
- It is a centroid-based algorithm, where each cluster is associated with a centroid.
- The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.
- The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters.
- The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from clusters.

Working of K-Means Algorithm

The working of the K-Means algorithm is explained in the below steps:

1. Choose the value of k and the k initial guesses for the centroids.
2. Compute the distance from each data point to each centroid.

   In two dimensions, the distance, d, between any two points, (X1, Y1) and (X2, Y2), in the Cartesian plane is typically expressed by using the Euclidean distance measure provided in

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

3. Compute the centroid, the center of mass, of each newly defined cluster from Step2.

   In two dimensions, the centroid (Xc, Yc) of the m points in a k-means cluster is calculated as follows in

$$(x_c, y_c) = \left(\frac{\sum_{i=1}^{m} x_i}{m}, \frac{\sum_{i=1}^{m} y_i}{m} \right)$$

4. **Calculate distance d, between pi and q**

   For a given point, pi, at (pi1, pi2, . . . pin) and a centroid, q, located at (q1, q2, . . . qn), the *distance, d, between pi and q*, is expressed as

$$d(p_i, q) = \sqrt{\sum_{j=1}^{n} (p_{ij} - q_j)^2}$$

5. **Calculate centroid, q**

The centroid, q, of a cluster of m points, (pi1, pi2, . . . pin) , is calculated as

$$(q_1, q_2, \ldots, q_n) = \left(\frac{\sum_{i=1}^{m} p_{i1}}{m}, \frac{\sum_{i=1}^{m} p_{i2}}{m}, \ldots, \frac{\sum_{i=1}^{m} p_{in}}{m} \right)$$

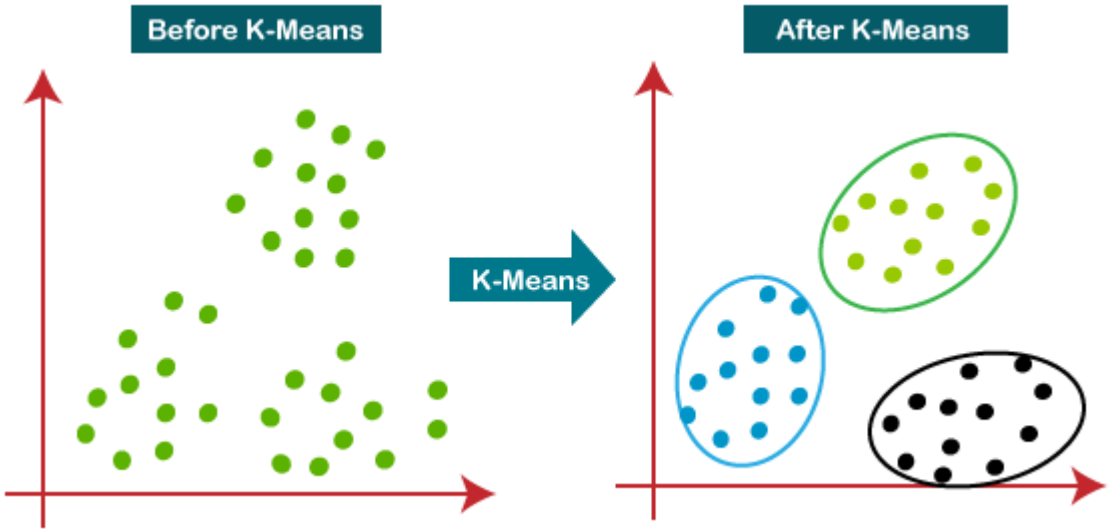


Figure 3.15 K-means Clustering Algorithm

7. Explain in detail about Instance based Learning and discuss KNN - k-Nearest Neighbor Learning?

Instance based learning:

- Instance-based learning are the systems that learn the training examples by heart and then generalizes to new instances based on some similarity measure.
- It is called instance-based because it builds the hypotheses from the training instances.
- It is also known as memory-based learning or lazy-learning.
- When a new instance is encountered, its relationship to the stored examples is examined in order to assign a target function value for the new instance.

- Instance-based methods are sometimes referred to as lazy learning methods because they delay processing until a new instance must be classified.
- A key advantage of lazy learning is that instead of estimating the target function once for the entire instance space, these methods can estimate it locally and differently for each new instance to be classified.
- Some of the instance-based learning algorithms are :
    - K Nearest Neighbor (KNN)
    - Self-Organizing Map (SOM)
    - Learning Vector Quantization (LVQ)
    - Locally Weighted Learning (LWL)
    - Case-Based Reasoning

k-Nearest Neighbor Learning - KNN

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Need of a K-NN Algorithm

- Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To

solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

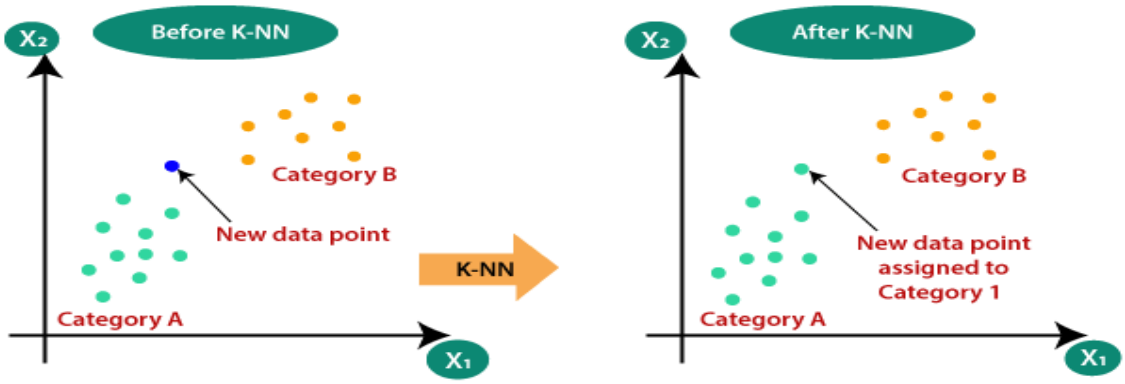


Figure 3.16 Explains the working of the K-NN Algorithm

Working of K-NN

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

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

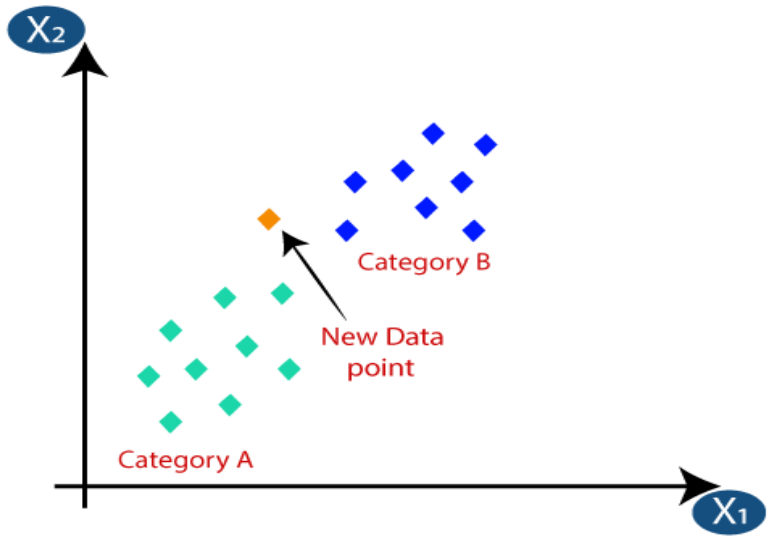


Fig.3.17 Euclidean Distance

- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points It can be calculated as:

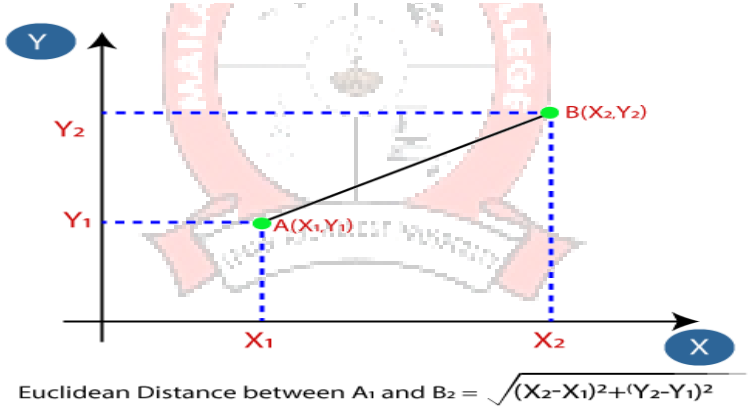


Fig.3.18 Euclidean Disatance

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

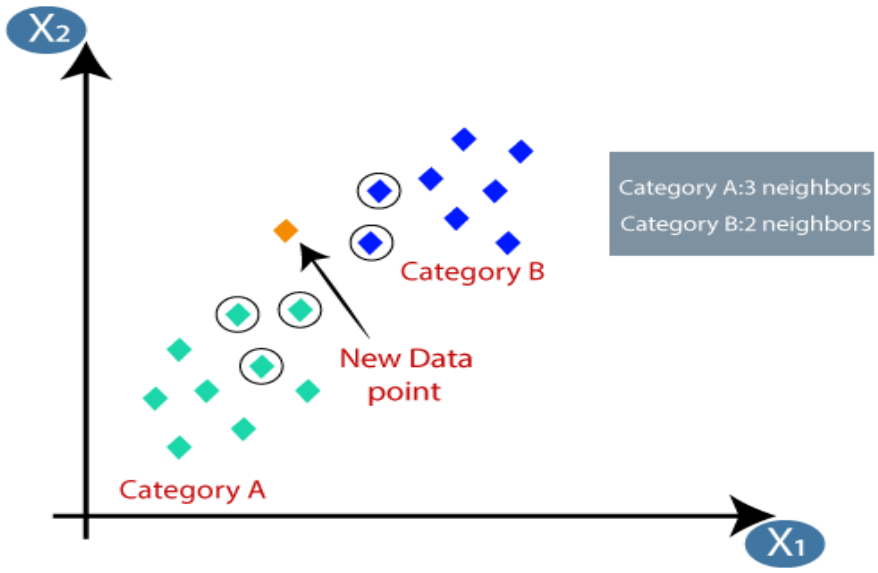


Figure 3.19 nearest neighbors

- Since 3 nearest neighbors are from category A in Figure 3.19, hence the new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

25. Explain in detail about Gaussian Mixture models and Expectation Maximization.

Explain the concept of Gaussian mixture model. [Nov 2024]

Discuss in detail about Expectation Maximization algorithm [Nov 2024]

Gaussian Mixture Model

- Gaussian mixture models (GMMs) are a type of machine learning algorithm. They are used to classify data into different categories based on the probability distribution.
- Gaussian mixture models can be used in many different areas, including finance, marketing and so much more.
- The Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mix of Gaussian distributions with unknown parameters.
- A Gaussian mixture model can be used for clustering, which is the task of grouping a set of data points into clusters.
- In general, K-means will be faster and more accurate when the data set is large and the clusters are well-separated. Gaussian mixture models will be more accurate when the data set is small or the clusters are not well-separated.
- GMM consists of two parts – mean vectors (μ) & covariance matrices (Σ).
- A Gaussian distribution is defined as a continuous probability distribution that takes on a bell-shaped curve. Another name for Gaussian distribution is the normal distribution.

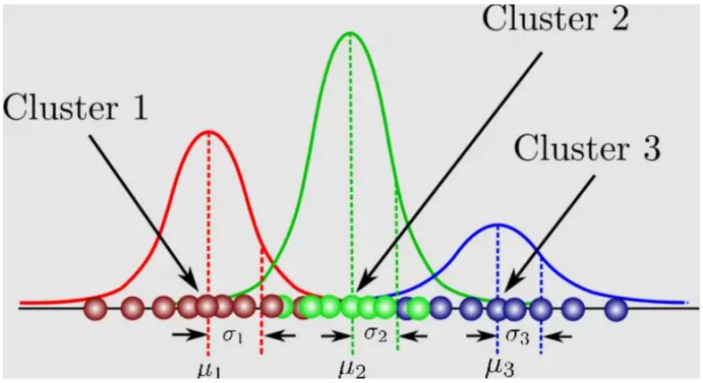


Fig.3.20 Gaussian Mixture Model

- Gaussian mixture models can handle missing data, whereas K-means cannot.
- The probability density function for one dimensional space of a Gaussian distribution is given by:

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

where μ is the mean and σ^2 is the variance.

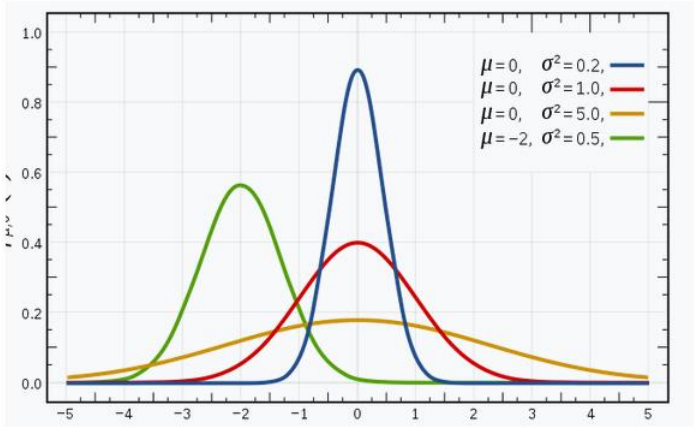


Fig. 3.20 2D Bell Shaped Curve

In the case of two variables, instead of a 2D bell-shaped curve, we will have a 3D bell curve as shown below:

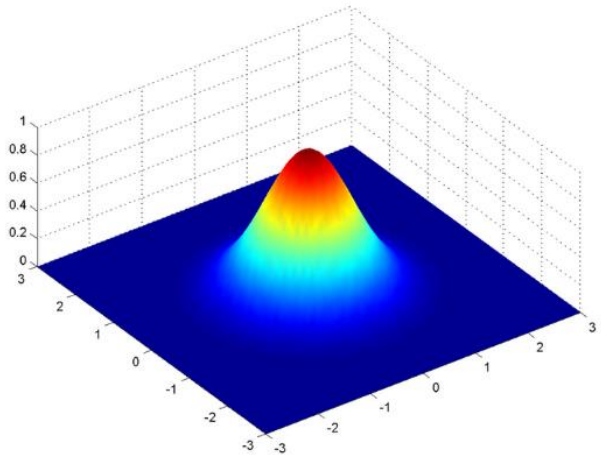


Fig.3.21 3D Bell Shaped Curve

The probability density function would be given by:

$$f(x \mid \mu, \Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left[-\frac{1}{2}(x-\mu)^t \Sigma^{-1}(x-\mu)\right]$$

where x is the input vector, μ is the 2D mean vector, and Σ is the 2×2 covariance matrix.

Thus, this multivariate Gaussian model would have x and μ as vectors of length d, and Σ would be a d x d covariance matrix.

Expectation-Maximization Algorithm

- The mean and variance value for each Gaussian distribution are determined using a technique called Expectation-Maximization (EM).
- Expectation-Maximization (EM) is a statistical algorithm for finding the right model parameters. Use EM when the data has missing values, or in other words, when the data is incomplete.
- These missing variables are called latent variables.
- Expectation-Maximization algorithm has two steps:
    - E-step: In this step, the available data is used to estimate (guess) the values of the missing variables
    - M-step: Based on the estimated values generated in the E-step, the complete data is used to update the parameters

- Expectation-Maximization is the base of many algorithms, including Gaussian Mixture Models.

E-step:

- For each point x_i, calculate the probability that it belongs to cluster/distribution c_1, c_2, … c_k. This is done using the below formula:

$$r_{ic} = \frac{\text{Probability Xi belongs to c}}{\text{Sum of probability Xi belongs to } c_1, c_2, .. c_k} = \frac{\pi_c \mathcal{N}(x_i \; ; \; \mu_c, \Sigma_c)}{\sum_{c'} \pi_{c'} \mathcal{N}(x_i \; ; \; \mu_{c'}, \Sigma_{c'})}$$

- This value will be high when the point is assigned to the right cluster and lower otherwise.

M-step:

1. The new density is defined by the ratio of the number of points in the cluster and the total number of points:

$$\prod = \frac{\text{Number of points assigned to cluster}}{\text{Total number of points}}$$

2. The mean and the covariance matrix are updated based on the values assigned to the distribution, in proportion with the probability values for the data point. Hence, a data point that has a higher probability of being a part of that distribution will contribute a larger portion:

$$\mu = \frac{1}{\text{Number of points assigned to cluster}} \sum_i r_{ic} x_i$$

$$\Sigma_c = \frac{1}{\text{Number of points assigned to cluster}} \sum_i r_{ic} (x_i - \mu_c)^T (x_i - \mu_c)$$

26. Compare Bagging, Boosting and Stacking ensemble methods. [Apr 2024]

- Ensemble learning is a machine learning methodology that combines predictions from multiple models. The method uses a weak learner model to produce the final prediction to improve the model performance. The ensemble concept is to combine individual weaker models and collectively become a strong learner.
- There are a few ensemble methods, but we would focus on the most popular and valuable:
  - Bagging
  - Boosting
  - Stacking

1. Bagging (Bootstrap Aggregating)

- **Objective**: Improve accuracy by reducing variance (i.e., overfitting).
- **How it works**:
  - Multiple base models are trained independently using different subsets of the training data.
  - These subsets are created by randomly sampling with replacement (bootstrap sampling).
  - Each model is trained on a different subset of the data and predictions are made by aggregating (e.g., majority voting for classification, averaging for regression) the results of these models.
- **Key Characteristics**:
  - **Parallelism**: Base models are trained independently.
  - **Independence**: No dependency between the base models.
  - **Examples**: Random Forest (a popular bagging method).
  - **Bias/Variance**: Reduces variance (less overfitting), but does not reduce bias much.
- **Advantages**:
  - Simple and easy to parallelize.
  - Works well with high variance models (like decision trees).

- **Disadvantages**:
    - Not effective for reducing bias (doesn't help if the base models are weak).
    - Might not perform well on certain types of data or models.

2. Boosting

- **Objective**: Improve accuracy by reducing bias (i.e., underfitting).
- **How it works**:
    - Sequentially trains base models, where each model corrects the errors made by the previous one.
    - Initially, all data points are weighted equally. After each model is trained, the misclassified points are given higher weights, so the next model focuses more on those difficult cases.
    - The final prediction is made by combining (usually through weighted voting or averaging) the predictions of all base models.
- **Key Characteristics**:
    - **Sequential**: Models are trained sequentially, each focusing on the errors of the previous one.
    - **Dependency**: There is a dependency between models.
    - **Examples**: AdaBoost, Gradient Boosting, XGBoost, LightGBM.
    - **Bias/Variance**: Reduces both bias and variance (very powerful for complex data).
- **Advantages**:
    - Often performs well on a wide range of problems.
    - Can handle both high bias and high variance issues.
    - Flexible and can be customized for different tasks.
- **Disadvantages**:
    - Sensitive to noisy data and outliers.
    - Slower than bagging due to the sequential nature of training.

- Can be prone to overfitting if not tuned properly (especially with too many rounds of boosting).

3. Stacking (Stacked Generalization)

- **Objective**: Improve accuracy by combining multiple different types of models to leverage their strengths.
- **How it works**:
  - Multiple different base models (often of different types, like decision trees, logistic regression, SVMs, etc.) are trained on the same dataset.
  - A "meta-model" (or a second-level model) is trained on the outputs of the base models. This meta-model learns how to combine the predictions of the base models to improve overall performance.
  - The final prediction is made by the meta-model, which takes the base models' predictions as input.
- **Key Characteristics**:
  - **Complexity**: Involves multiple different models and a meta-model.
  - **Layered**: First level models (base models) are trained independently, then a meta-model is trained on their predictions.
  - **Examples**: Multi-layered models combining various classifiers/regressors like a meta-model on Random Forest, XGBoost, and Logistic Regression.
  - **Bias/Variance**: Can reduce both bias and variance, depending on the combination of base models and meta-model.
- **Advantages**:
  - Often leads to better performance than any individual model.
  - Can combine a diverse set of models to take advantage of their different strengths.
- **Disadvantages**:
  - Computationally expensive and complex to implement.

- Requires careful selection of base models and meta-model to avoid overfitting.
- May not be easy to interpret.

Summary Table:

Feature	Bagging	Boosting	Stacking
Goal	Reduce variance (overfitting)	Reduce bias (underfitting)	Combine diverse models for better accuracy
Model Training	Independent parallel models	Sequential, correcting errors of previous models	Independent base models, meta-model combines outputs
Data Subsampling	Yes, bootstrapped samples	No, uses all data each time	No, all base models use full dataset
Model Dependence	Independent	Dependent	Dependent (meta-model depends on base models)
Aggregation Method	Majority voting or averaging	Weighted majority voting or weighted averaging	Meta-model combines base model predictions
Bias/Variance	Reduces variance	Reduces bias and variance	Reduces both bias and variance (depending on models)
Examples	Random Forest	AdaBoost, Gradient Boosting, XGBoost	Random Forest + XGBoost + Logistic Regression (as meta-model)
Advantages	Simple, easy to parallelize	Works well with complex datasets, reduces both bias and	Can outperform individual models, combines

Feature	Bagging	Boosting	Stacking
		variance	strengths
Disadvantages	Limited by the base model's bias	Sensitive to noisy data, prone to overfitting	Complex, computationally expensive

Key Differences:

- **Bagging** focuses on reducing variance by training models independently and combining their predictions, making it suitable for high variance, low bias models like decision trees.
- **Boosting** works sequentially, where each new model corrects the mistakes of the previous one, making it powerful for improving underperforming models and reducing both bias and variance.
- **Stacking** combines multiple models of different types, with a meta-model learning how to optimally combine them, leading to strong predictive performance but at the cost of increased complexity.

Each method is useful depending on the problem at hand. Bagging is great for reducing variance, boosting is powerful when you need to improve the model's accuracy (especially in terms of bias), and stacking is ideal when you want to leverage the strengths of different types of models.

27. Cluster the following eight points (with (x,y) representing locations) into three clusters using K-Means clustering method. A1(2,10), A2(2,5), A3(8,4), A4(5,8),A6(6,4),A7(1,2),A8(4,9). [Apr 2024]

To cluster the given eight points into three clusters using the **K-Means clustering method**, we'll follow these steps:

Steps in K-Means Clustering

1. **Initialize**: Select three initial centroids randomly from the given points.
2. **Assignment**: Assign each point to the closest centroid.
3. **Update**: Compute the new centroids by averaging the points in each cluster.

4. **Repeat**: Continue reassigning points and updating centroids until convergence.

Given Points

Point Coordinates (x, y)

A1 (2,10)

A2 (2,5)

A3 (8,4)

A4 (5,8)

A5 (7,5)

A6 (6,4)

A7 (1,2)

A8 (4,9)

Step 1: Choose Initial Centroids (Random Selection)

We will randomly select three points as initial centroids. Let's take:

- **C1 = A1 (2,10)**
- **C2 = A4 (5,8)**
- **C3 = A7 (1,2)**

Step 2: Assign Each Point to the Nearest Centroid

We calculate the Euclidean distance from each point to the centroids and assign them to the closest cluster.

Distance=(x2−x1)2+(y2−y1)2\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}Distance=(x2−x1)2+(y2−y1)2

We'll compute these distances programmatically and determine the initial clusters. Let me run these calculations.

Initial Cluster Assignment

After calculating distances, the points are assigned as follows:

- **Cluster C1 (Centroid: A1 (2,10))**:
  - A1 (2,10)
- **Cluster C2 (Centroid: A4 (5,8))**:
  - A3 (8,4), A4 (5,8), A5 (7,5), A6 (6,4), A8 (4,9)
- **Cluster C3 (Centroid: A7 (1,2))**:
  - A2 (2,5), A7 (1,2)

Step 3: Compute New Centroids

To update the centroids, we calculate the mean of the points in each cluster.

New Centroid=(∑xn,∑yn)\text{New Centroid} = \left(\frac{\sum x}{n}, \frac{\sum y}{n} \right)New Centroid=(n∑x,n∑y)

Let's compute the new centroids.

Updated Centroids

After recomputing the centroids based on the current cluster assignments:

- **New Centroid C1**: (2.0, 10.0) (Remains the same)
- **New Centroid C2**: (6.0, 6.0)
- **New Centroid C3**: (1.5, 3.5)

Step 4: Repeat Assignment with Updated Centroids

Now, we reassign each point based on the new centroids. Let’s compute this next.

Updated Cluster Assignment

After reassigning points based on the new centroids:

- **Cluster C1 (Centroid: (2.0, 10.0))**:
  - A1 (2,10), A8 (4,9)
- **Cluster C2 (Centroid: (6.0, 6.0))**:
  - A3 (8,4), A4 (5,8), A5 (7,5), A6 (6,4)
- **Cluster C3 (Centroid: (1.5, 3.5))**:

- A2 (2,5), A7 (1,2)

Step 5: Check for Convergence

The cluster assignments have changed slightly. To finalize, we compute new centroids again and check if they remain the same. If so, we stop; otherwise, we repeat. Let's verify.

Step 6: Final Check for Convergence

The new centroids after the second iteration:

- **Final Centroid C1**: (3.0, 9.5)
- **Final Centroid C2**: (6.5, 5.25)
- **Final Centroid C3**: (1.5, 3.5)

Since the centroids have changed again, the process would typically continue until convergence. However, based on these adjustments, the clusters are stabilizing.

28. Illustrate the principles of K-means algorithms with an example.

[Nov 2024]

- K-Means is a clustering algorithm that groups data into K clusters. It follows an iterative approach to find the best cluster centers.

Step-by-Step Example

- Let's consider a dataset with the following **six points** in a **2D space**:

Point Coordinates (x, y)

A1 (2, 3)

A2 (5, 4)

A3 (9, 6)

A4 (4, 7)

A5 (8, 1)

Point Coordinates (x, y)

A6 (7, 3)

We will apply **K-Means** with K=2K = 2K=2 clusters.

Step 1: Initialize Centroids

We randomly select **two** initial cluster centers. Let's pick:

- **C1 = (2,3) (A1)**
- **C2 = (8,1) (A5)**

Step 2: Assign Each Point to the Closest Centroid

We calculate the **Euclidean distance** from each point to the centroids:

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

Let's compute the distances and assign each point to the nearest centroid.

Let me calculate and display the assignments.

Initial Cluster Assignment

After computing the distances, the points are assigned as follows:

- **Cluster C1 (Centroid: (2,3))**:
  - A1 (2,3)
  - A2 (5,4)
  - A4 (4,7)
- **Cluster C2 (Centroid: (8,1))**:
  - A3 (9,6)
  - A5 (8,1)
  - A6 (7,3)

Step 3: Compute New Centroids

We compute the new centroids by averaging the coordinates of the points in each cluster:

$$\text{New Centroid} = \left(\frac{\sum x}{n}, \frac{\sum y}{n} \right)$$

Let's compute the new centroids.

Updated Centroids

After recomputing the centroids:

- **New Centroid C1**: (3.0, 7.67)
- **New Centroid C2**: (7.0, 4.33)

Step 4: Repeat Assignment

Now, we reassign each point to the new centroids and check if clusters have changed. Let's do this.

Updated Cluster Assignment

After reassigning based on the new centroids:

- **Cluster C1 (Centroid: (3.0, 7.67))**:
  - A1 (2,3)
  - A4 (4,7)
- **Cluster C2 (Centroid: (7.0, 4.33))**:
  - A2 (5,4)
  - A3 (9,6)
  - A5 (8,1)
  - A6 (7,3)

Since the cluster assignments changed, we would repeat the process until convergence (i.e., when centroids no longer move).

UNIT IV - NEURAL NETWORKS

Perceptron - Multilayer perceptron, activation functions, network training – gradient descent optimization – stochastic gradient descent, error backpropagation, from shallow networks to deep networks –Unit saturation (aka the vanishing gradient problem) – ReLU, hyperparameter tuning, batch normalization, regularization, dropout.

PART A

1. Differentiate between Single layer and Multilayer perceptron. [Apr 2024]

Feature	Single-Layer Perceptron (SLP)	Multi-Layer Perceptron (MLP)
Architecture	Contains only **one** layer of neurons (input → output).	Contains **multiple hidden layers** between input and output.
Complexity	Simple, only capable of learning **linearly separable** problems.	More complex, can learn **non-linear** relationships.
Hidden Layers	No hidden layer, only input and output layers.	Has one or more **hidden layers** for deeper learning.
Computation Power	Limited computational capability.	More powerful, used in deep learning.
Activation Function	Usually uses a step function or linear function.	Uses non-linear activation functions like ReLU, Sigmoid, or Tanh.
Training Algorithm	Uses **Perceptron Learning Algorithm** (PLA).	Uses **Backpropagation** and Gradient Descent.
Examples	Used for basic classification problems (e.g., AND, OR logic gates).	Used for complex tasks like image recognition, NLP, etc.

2. List the problems associated with Backpropagation Neural network. [Apr 2024]

Problems Associated with Backpropagation in Neural Networks

Backpropagation is a widely used algorithm for training neural networks, but it has several challenges:

1. Slow Convergence
2. Vanishing Gradient Problem
3. Exploding Gradient Problem
4. Local Minima and Saddle Points

5. Overfitting
6. Need for Large Amounts of Data
7. High Computational Cost
8. Choice of Hyperparameters
9. Difficulty in Training Deep Networks
10. Non-Convex Loss Function

3. List different activation function used in neural networks. [Nov 2024]

1. ReLU Function
2. Sigmoid Function
3. Linear Function
4. Tanh Function
5. Softmax Function

4. What is vanishing gradient problem? [Nov 2024]

- The vanishing gradient problem is an issue that sometimes arises when training machine learning algorithms through gradient descent.
- This most often occurs in neural networks that have several neuronal layers such as in a deep learning system, but also occurs in recurrent neural networks.

5. Define neuron.

- A neuron is a cell in the brain whose principal function is the collection, processing, and dissemination of electrical signals.

2. Define neural networks.

- The brain's information-processing capacity is thought to emerge primarily from *networks* of such neurons. For this reason, some of the earliest AI work aimed to create artificial neural networks.

3. What are the two main categories of neural network structures?

- acyclic or feed-forward net-works

- cyclic or recurrent networks

4. Define Perceptron.

- A network with all the inputs connected directly to the outputs is called a single layer neural network, or a perceptron network.
- Since each output unit is independent of the others-each weight affects only one of the outputs.

5. Define Multi Layer Perceptron.

- Multi-layer perception is also known as MLP. It is fully connected dense layers, which transform any input dimension to the desired dimension.
- A multi-layer perception is a neural network that has multiple layers. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons.
- A multi-layer perceptron has one input layer and for each input, there is one neuron (or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes. A schematic diagram of a Multi-Layer Perceptron (MLP).

6. Define Activation function.

- In an artificial neural network, the function which takes the incoming signals as input and produces the output signal is known as the activation function.

7. List down the advantages of multilayer perceptron.

- It can be used to solve complex nonlinear problems.
- It handles large amounts of input data well.
- Makes quick predictions after training.
- The same accuracy ratio can be achieved even with smaller samples.

8. How do you train a neural model?

1. First an ANN will require a **random weight initialization**
2. Split the dataset in **batches (batch size)**

3. Send the batches 1 by 1 to the GPU
4. Calculate the **forward pass** (what would be the output with the current weights)
5. Compare the calculated output to the expected output **(loss)**
6. Adjust the **weights** (using the **learning rate** increment or decrement) according to the **backward pass (backward gradient propagation)**.
7. Go back to step 2.

9. List out the activation functions.

6. ReLU Function
7. Sigmoid Function
8. Linear Function
9. Tanh Function
10. Softmax Function

10. Define Gradient Descent Optimization.

- Gradient Descent is a generic optimization algorithm capable of finding optimal solutions to a wide range of problems.
- The general idea is to tweak parameters iteratively in order to minimize the cost function.
- An important parameter of Gradient Descent (GD) is the size of the steps, determined by the learning rate hyperparameters. If the learning rate is too small, then the algorithm will have to go through many iterations to converge, which will take a long time, and if it is too high we may jump the optimal value.

11. List the types of Gradient Descent.

- 1.Batch Gradient Descent
- 2.Stochastic Gradient Descent
- 3.Mini-batch Gradient Descent

12. Define Stochastic Gradient Descent (SGD).

- In Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration.
- In Gradient Descent, there is a term called "batch" which denotes the total number of samples from a dataset that is used for calculating the gradient for each iteration.
- In typical Gradient Descent optimization, like Batch Gradient Descent, the batch is taken to be the whole dataset.

13. What are the advantages and disadvantages of SGD?

Advantages:

- **Speed:** SGD is faster than other variants of Gradient Descent.
- **Memory Efficiency:**it is memory-efficient and can handle large datasets that cannot fit into memory.
- **Avoidance of Local Minima:** Due to the noisy updates in SGD, it has the ability to escape from local minima and converge to a global minimum.

Disadvantages:

- **Noisy updates:** The updates in SGD are noisy and have a high variance, which can make the optimization process less stable and lead to oscillations around the minimum.
- **Slow Convergence:** SGD may require more iterations to converge to the minimum since it updates the parameters for each training example one at a time.
- **Sensitivity to Learning Rate:** The choice of learning rate can be critical in SGD since using a high learning rate can cause the algorithm to overshoot the minimum, while a low learning rate can make the algorithm converge slowly.
- **Less Accurate:** Due to the noisy updates, SGD may not converge to the exact global minimum and can result in a suboptimal solution. This can be mitigated by using techniques such as learning rate scheduling and momentum-based updates.

14. Define Backpropagation.

- The neural network has neurons that work in correspondence with *weight, bias,* and their respective activation function. In a neural network, we would update the weights and biases of the neurons on the basis of the error at the output. This process is known as backpropagation**.**

15. List out the types of Backpropagations.

- Two Types of Backpropagation Networks are:
    1. Static Back-propagation
    2. Recurrent Backpropagation

- **Static back-propagation:**
    - It is one kind of backpropagation network which produces a mapping of a static input for static output. It is useful to solve static classification issues like optical character recognition.
- **Recurrent Backpropagation:**
    - Recurrent Back propagation in data mining is fed forward until a fixed value is achieved. After that, the error is computed and propagated backward.

14. What are the advantages and disadvantages of error backropagation?

Advantages:

- It does not have any parameters to tune except for the number of inputs.
- It is highly adaptable and efficient and does not require any prior knowledge about the network.
- It is a standard process that usually works well.

Disadvantages:

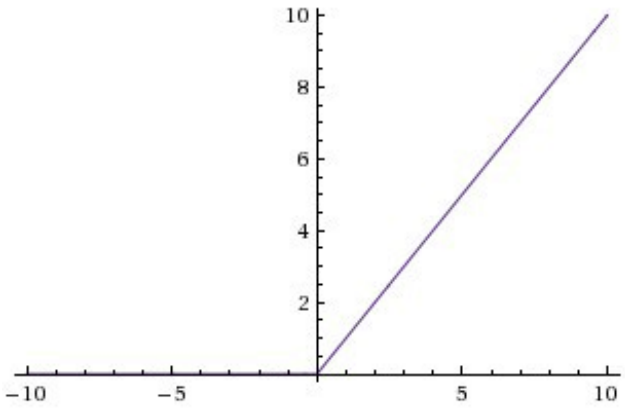
- The performance of backpropagation relies very heavily on the training data.
- Backpropagation needs a very large amount of time for training.
- Backpropagation requires a matrix-based method instead of mini-batch.

15. Define Unit saturation (aka the vanishing gradient problem).

- The vanishing gradient problem is an issue that sometimes arises when training machine learning algorithms through gradient descent. This most often occurs in neural networks that have several neuronal layers such as in a deep learning system, but also occurs in recurrent neural networks.
- The key point is that the calculated partial derivatives used to compute the gradient as one goes deeper into the network. Since the gradients control how much the network learns during training, the gradients are very small or zero, then little to no training can take place, leading to poor predictive performance.

16. Define Rectified linear unit (ReLU).

- It Stands for *Rectified linear unit*. It is the most widely used activation function. Chiefly implemented in *hidden layers* of Neural network. (see the below figure).
- **Equation :-** ***A(x) = max(0,x)***. It gives an output x if x is positive and 0 otherwise.
- **Value Range :-** [0, inf)
- **Nature :-** non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.



ReLU Function

- **Uses :-** ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few

neurons are activated making the network sparse making it efficient and easy for computation.

- In simple words, RELU learns *much faster* than sigmoid and Tanh function.

17. Define Normalization.

- Normalization is a data pre-processing tool used to bring the numerical data to a common scale without distorting its shape.

18. Define Batch Normalization.

- Batch normalization is a process to make neural networks faster and more stable through adding extra layers in a deep neural network.
- The new layer performs the standardizing and normalizing operations on the input of a layer coming from a previous layer.
- A typical neural network is trained using a collected set of input data called batch**.**

19. Define Hyperparameter tuning.

- A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. By training a model with existing data, we are able to fit the model parameters.
- However, there is another kind of parameter, known as Hyperparameters**,** that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.

20. What are the two strategies of Hyperparameter tuning?

- GridSearchCV
- RandomizedSearchCV

21. Define GridSearchCV.

- In GridSearchCV approach, the machine learning model is evaluated for a range of hyperparameter values.

- This approach is called GridSearchCV, because it searches for the best set of hyperparameters from a grid of hyperparameters values.

22. Define RandomizedSearchCV.

- RandomizedSearchCV solves the drawbacks of GridSearchCV, as it goes through only a fixed number of hyperparameter settings.
- It moves within the grid in a random fashion to find the best set of hyperparameters.
- This approach reduces unnecessary computation.

23. Define Overfitting.

- When a model performs well on the training data and does not perform well on the testing data, then the model is said to have high generalization error.
- In other words, in such a scenario, the model has low bias and high variance and is too complex. This is called overfitting.

24. What is Regularization?

- Regularization is one of the most important concepts of machine learning.
- It is a technique to prevent the model from overfitting by adding extra information to it. Regularization helps choose a simple model rather than a complex one.

25. List out the Regularization Technique.

1. Hints
2. Weight decay:
3. Ridge Regression (or) L2 Regularization.
4. Lasso Regression (or) L1 Regularization.
5. Dropout

26. Define L1 Regularization and L2 Regularization.

Lasso Regression (or) L1 Regularization.

- Least Absolute Shrinkage and Selection Operator (or LASSO) Regression penalizes the coefficients to the extent that it becomes zero.
- It eliminates the insignificant independent variables. This regularization technique uses the L1 norm for regularization.

$$E' = E + \frac{\lambda}{2}\sum_i |w_i|$$

Ridge Regression or L2 Regularization

- The Ridge regression technique is used to analyze the model where the variables may be having multicollinearity.
- It reduces the insignificant independent variables though it does not remove them completely. This type of regularization uses the L_2 norm for regularization.

$$E' = E + \frac{\lambda}{2}\sum_i w_i^2$$

27. Define Dropout.

- "Dropout" in machine learning refers to the process of randomly ignoring certain nodes in a layer during training.

28. Difference between Shallow and Deep neural network.

Shallow Network	Deep Network
A shallow neural network has only one hidden layer between the input and output layers	A deep neural network has multiple hidden layers
A shallow network might be used for simple tasks like image classification.	A deep network might be used for more complex tasks like image segmentation or natural language processing.

A shallow network is that it is computationally less expensive to train, and can be sufficient for simple tasks	It is more computationally expensive to train and may require more data to avoid overfitting.
It may not be powerful enough to capture complex patterns in the data.	It capture more complex patterns in the data and potentially achieve higher accuracy.

29. Difference between Stochastic Gradient Descent and Gradient Descent.

Stochastic Gradient Descent	Gradient Descent
Computes gradient using the whole Training sample	Computes gradient using a single Training sample
Not suggested for huge training samples.	Can be used for large training samples.
Gives optimal solution given sufficient time to converge.	Gives good solution but not optimal.
Deterministic in nature.	Stochastic in nature.
Convergence is slow.	Reaches the convergence much faster.

30. What is meant by Training set?

- Training set is a set of pairs of input patterns with corresponding desired output patterns.
- Each pair represents how the network is supposed to respond to a particular input.
- The network is trained to respond correctly to each input pattern from the training set.

31. What is meant by Test set?

- The test set is the dataset that the model is trained on.

32. Difference between Data Mining and Machine learning.

Data Mining	Machine Learning
Extracting useful information from large amount of data	Introduce algorithm from data as well as from past experience
Used to understand the data flow	Teaches the computer to learn and understand from the data flow
Human interference is more in it.	No human effort required after design
Huge databases with unstructured data	Existing data as well as algorithms
Historical data	Historical and real-time data
Data mining is more of a research using methods like machine learning	Self learned and trains system to do the intelligent task

33. Define Forward Pass.

- Forward Propagation is the way to move from the Input layer (left) to the Output layer (right) in the neural network.
- A neural network can be understood by a collection of connected input/output nodes.

34. Define Backward Pass.

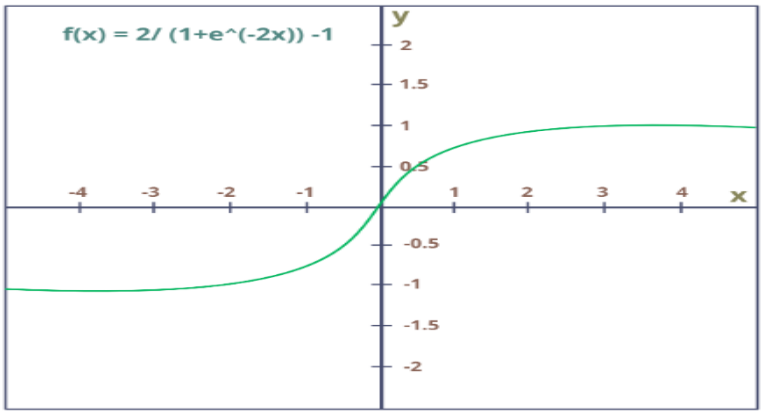
- In the backward pass, the flow is reversed so that we start by propagating the error to the output layer until reaching the input layer passing through the hidden layer(s).
- The process of propagating the network error from the output layer to the input layer is called backward propagation, or simple backpropagation.

35. Define Tanh Function.

- The activation that works almost always better than sigmoid function is Tanh function also known as Tangent Hyperbolic function. It's actually mathematically shifted version of the sigmoid function. Both are similar and can be derived from each other.
- **Equation :-**

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

- **Value Range :-** -1 to +1
- **Nature :-** non-linear
- **Uses :-** Usually used in hidden layers of a neural network as it's values lies between **-1 to 1** hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in *centering the data* by bringing mean close to 0. This makes learning for the next layer much easier.

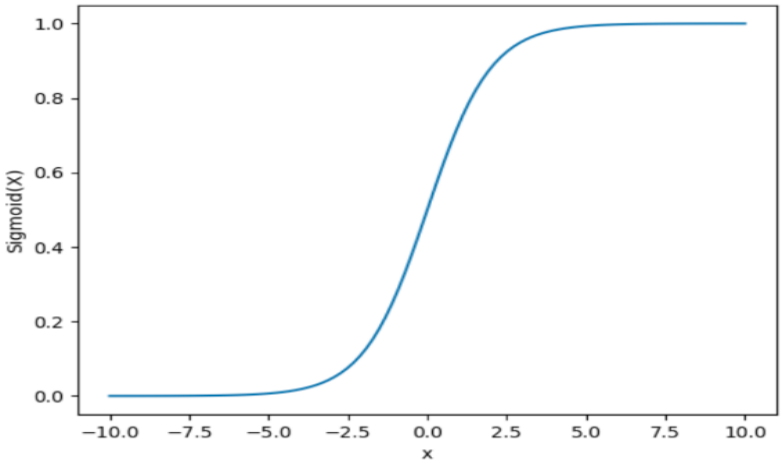


Tanh Function

36. Define Sigmoid Function.

- It is a function which is plotted as **'S'** shaped graph.
- **Equation :** $A = 1/(1 + e^{-x})$
- **Nature :** Non-linear. Notice that X values lies between -2 to 2, Y values are very steep. This means, small changes in x would also bring about large changes in the value of Y.

- **Value Range :** 0 to 1
- **Uses :** Usually used in output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to be ***1*** if value is greater than **0.5** and ***0*** otherwise.



37. What is Leaky ReLU?

- In the *leaky ReLU* (the output is also linear on the negative side but with a smaller slope, just enough to make sure that there will be updates for negative activations, albeit small:

$$\text{leaky-ReLU}(a) = \begin{cases} a & \text{if } a > 0 \\ \alpha a & \text{otherwise} \end{cases}$$

38. What is Artificial Neuron?

- An artificial neuron is a connection point in an artificial neural network.
- Artificial neural networks, like the human body's biological neural network, have a layered architecture and each network node (connection point) has the capability to process input and forward output to other nodes in the network.

39. What is meant by Feed forward neural network?

- "The process of receiving an input to produce some kind of output to make some kind of prediction is known as Feed Forward."

- Feed Forward neural network is the core of many other important neural networks such as convolution neural network.

40. Define Bias.

- Neural network bias can be defined as the constant which is added to the product of features and weights. It is used to offset the result. It helps the models to shift the activation function towards the positive or negative side.

PART B

1. Explain in detail about Perceptron.

- The *perceptron* is the basic processing element.
- It has inputs that may come from the environment or may be the outputs of other perceptrons.
- Associated with each input, $x_j \in \Re$, $j = 1, \ldots, d$, is a *connection weight*, or *synaptic weight* $w_j \in \Re$, and the output, y, in the simplest case is a weighted sum of the inputs **(see figure 4.1)**

$$y = \sum_{j=1}^{d} w_j x_j + w_0 \longrightarrow 1$$

- w_0 is the intercept value to make the model more general

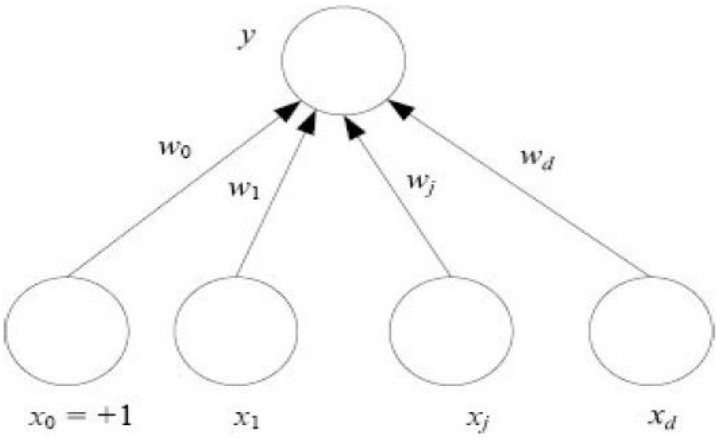


Figure 4.1 Simple Perceptron

- We can write the output of the perceptron as a dot product

$$\mathbf{Y = w^T x}$$

- where $w = [w_0, w_1, \ldots, w_d]^T$ and $x = [1, x_1, \ldots, x_d]^T$ are *augmented* vectors to include also the bias weight and input.
- During testing, with given weights, w, for input x, we compute the output y. To implement a given task, we need to *learn* the weights w, the parameters of the system, such that correct outputs are generated given the inputs.

- When $d = 1$ and x is fed from the environment through an input unit, we have

$$y = w x + w_0$$

where w as the slope and w_0 as the intercept

- Thus this perceptron with one input and one output can be used to implement a linear fit. With more than one input, the line becomes a (hyper)plane, and the perceptron with more than one input can be used to implement multivariate linear fit.
- The perceptron as defined in equation 1 defines a hyperplane and as such can be used to divide the input space into two:
  - the half-space where it is positive and the half-space where it is negative
- By using it to implement a linear discriminant function, the perceptron can separate two classes by checking the sign of the output. If we define $s(\cdot)$ as the *threshold function*

$$s(a) = \begin{cases} 1 & \text{if } a > 0 \\ 0 & \text{otherwise} \end{cases}$$

then we can,

$$\text{choose} \begin{cases} C_1 & \text{if } s(\boldsymbol{w}^T \boldsymbol{x}) > 0 \\ C_2 & \text{otherwise} \end{cases}$$

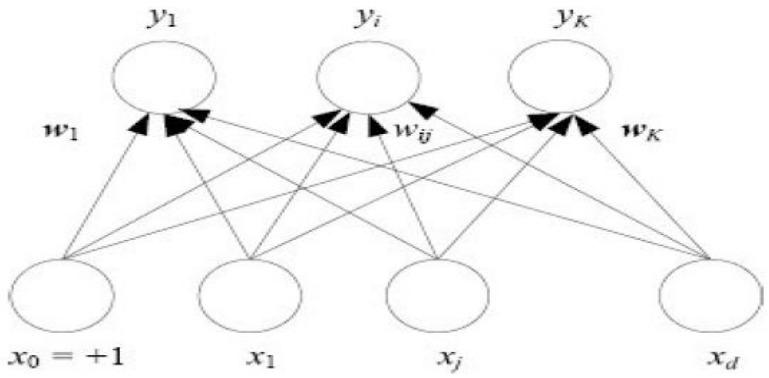


Figure 4.2 *K* parallel perceptrons

From the Figure 4.2,

- x_j, $j = 0, \ldots, d$ are the inputs and y_i, $i = 1, \ldots, K$ are the outputs. w_{ij} is the weight of the connection from input xj to output yi. Each output is a weighted sum of the inputs. When used for a K-class classification problem, there is a postprocessing to choose the maximum, or softmax if we need the posterior probabilities.
- it is assumed that a hyperplane $w^T x = 0$ can be found that separates $x^t \in C1$ and $x^t \in C2$. If at a later stage we need the posterior probability.
- Example, to calculate risk—we need to use the sigmoid function at the output as

$$o = w^T x$$
$$y = \text{sigmoid}(o) = \frac{1}{1 + \exp[-w^T x]}$$

- When there are $K > 2$ outputs, there are K perceptrons, each of which has a weight vector wi (see figure 5.2)

$$y_i = \sum_{j=1}^{d} w_{ij} x_j + w_{i0} = w_i^T x$$
$$y = Wx$$

- where w_{ij} is the weight from input x_j to output y_i. W is the $K \times (d + 1)$ weight matrix of w_{ij} whose rows are the weight vectors of the K perceptrons. When used for classification, during testing, we

$$\text{choose } C_i \text{ if } y_i = \max_k y_k$$

- Each perceptron is a *local* function of its inputs and synaptic weights. In classification, if we need the posterior probabilities and use the softmax, we need the values of all outputs.
- Implementing this as a neural network results in a two-stage process, where the first calculates the weighted sums, and the second calculates the softmax values; but we denote this as a single layer:

$$o_i = \mathbf{w}_i^T \mathbf{x}$$
$$y_i = \frac{\exp o_i}{\sum_k \exp o_k}$$

- a linear transformation from a d dimensional space to a K-dimensional space and can also be used for dimensionality reduction if $K < d$. we have a two-layer network where the first layer of perceptrons implements the linear transformation and the second layer implements the linear regression or classification in the new space.

2. Explain in detail about Multi Layer Perceptron.

Describe in brief about Multilayer perceptron activation functions. [Apr 2024]

- Multi-layer perception is also known as MLP. It is fully connected dense layers, which transform any input dimension to the desired dimension.
- A multi-layer perception is a neural network that has multiple layers. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons.
- A multi-layer perceptron has one input layer and for each input, there is one neuron (or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes. A schematic diagram of a Multi-Layer Perceptron (MLP) is shown in **Figure 4.3**.

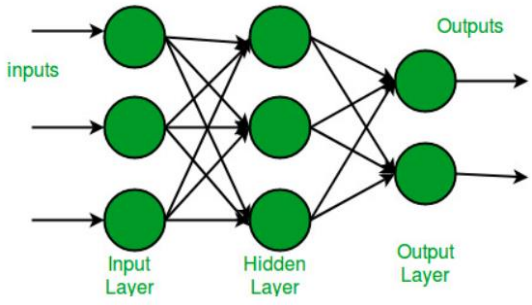


Figure 4.3 A Multi-Layer Perceptron (MLP)

- In the multi-layer perceptron diagram, we can see that there are three inputs and thus three input nodes and the hidden layer has three nodes.
- The output layer gives two outputs, therefore there are two output nodes. The nodes in the input layer take input and forward it for further process, in the diagram above the nodes in the input layer forwards their output to each of the three nodes in the hidden layer, and in the same way, the hidden layer processes the information and passes it to the output layer.
- Every node in the multi-layer perception uses a sigmoid activation function. The sigmoid activation function takes real values as input and converts them to numbers between 0 and 1 using the sigmoid formula.

$$(x)=1/(1+(\exp(-x))$$

- A perceptron that has a single layer of weights can only approximate linear functions of the input and cannot solve problems like the XOR, where the discrimininant to be estimated is nonlinear.
- Similarly, a perceptron cannot be used for nonlinear regression. This limitation does not apply to feedforward networks with an intermediate or *hidden layer* between the input and the output layers.
- If used for classification, such *multilayer perceptrons* (MLP) can implement nonlinear discriminants and, if used for regression, can approximate nonlinear functions of the input.

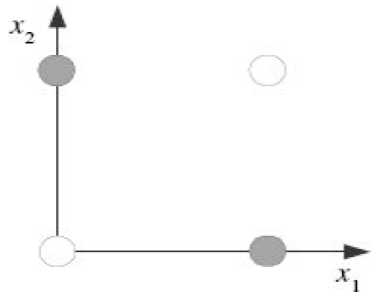


Figure 4.4 XOR problem is not linearly separable.

From the Figure 4.4, We cannot draw a line where the empty circles are on one side and the filled circles on the other side.

- Input x is fed to the input layer (including the bias), the "activation" propagates in the forward direction, and the values of the hidden units z_h are calculated **(see Figure 4.5).** Each hidden unit is a perceptron by itself and applies the nonlinear sigmoid function to its weighted sum:

$$z_h = \text{sigmoid}(\boldsymbol{w}_h^T \boldsymbol{x}) = \frac{1}{1 + \exp\left[-\left(\sum_{j=1}^{d} w_{hj} x_j + w_{h0}\right)\right]}, \quad h = 1, \ldots, H$$

- The output yi are perceptrons in the second layer taking the hidden units as their inputs

$$y_i = \boldsymbol{v}_i^T \boldsymbol{z} = \sum_{h=1}^{H} v_{ih} z_h + v_{i0}$$

- Where there is also a bias unit in the hidden layer, which we denote by z_0, and v_{i0} are the bias weights. The input layer of x_j is not counted since no computation is done there and when there is a hidden layer, this is a two-layer network.
- In a two-class discrimination task, there is one sigmoid output unit and when there are $K > 2$ classes, there are K outputs with softmax as the output nonlinearity.

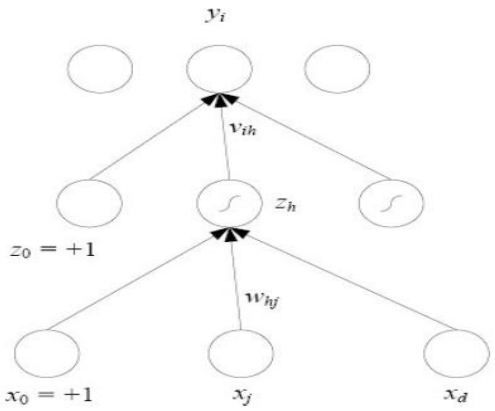


Figure 4.5 The structure of a multilayer perceptron.

- **From the Figure 4.5** x_j, $j = 0,\dots, d$ are the inputs and z_h, $h = 1, \dots, H$ are the hidden units where H is the dimensionality of this hidden space. z_0 is the bias of the hidden layer. y_i, $i = 1, \dots, K$ are the output units. w_{hj} are weights in the first layer, and v_{ih} are the weights in the second layer.

- The MLP that implements XOR with two hidden units that implement the two ANDs and the output that takes an OR of them.

x_1	x_2	z_1	z_2	y
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

- One is not limited to having one hidden layer, and more hidden layers with their own incoming weights can be placed after the first hidden layer with sigmoid hidden units, thus calculating nonlinear functions of the first layer of hidden units and implementing more complex functions of the inputs.

3. Explain in Detail about Activation function.

- Artificial neurons are elementary units in an artificial neural network. The artificial neuron receives one or more inputs and sums them to produce an output. Each input is separately weighted, and the sum is passed through a function known as an activation function or transfer function.

- In an artificial neural network, the function which takes the incoming signals as input and produces the output signal is known as the activation function.

- Neuron should be activated or not by calculating the weighted sum and further adding bias to it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

- The neural network has neurons that work in correspondence with *weight, bias,* and their respective activation function. In a neural network, we would

update the weights and biases of the neurons on the basis of the error at the output. This process is known as **Backpropagation**.

- Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases.

Need for Non-linear activation function

- A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.
- The two main categories of activation functions are:

1. Linear Activation Function
2. Non-linear Activation Functions

Linear Activation Function

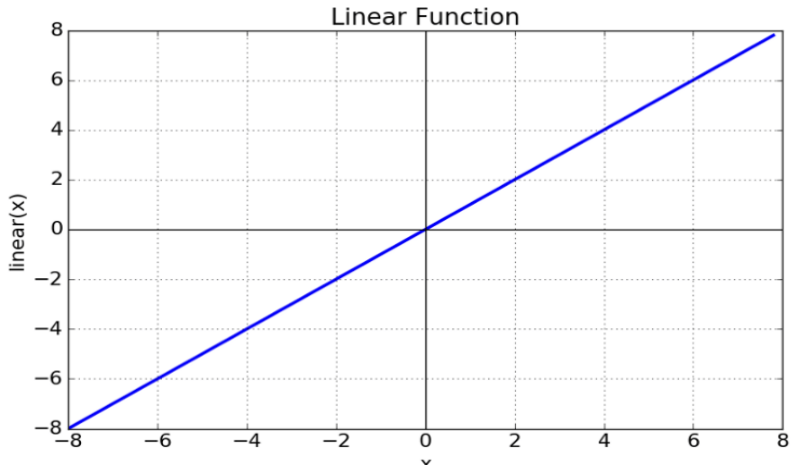


Figure 4.6 Linear Activation Function

Non-linear Activation Functions

1. **Linear Function**

- Linear function has the equation similar to as of a straight line i.e. **y = x (see in Figure 4.6).**

- No matter how many layers we have, if all are linear in nature, the final activation function of last layer is nothing but just a linear function of the input of first layer.
- **Range :** -inf to +inf
- **Uses : Linear activation function** is used at just one place i.e. output layer.

2. Sigmoid Function

- It is a function which is plotted as **'S'** shaped graph **(Refer Figure 4.7)** .
- **Equation :** $A = 1/(1 + e^{-x})$
- **Nature :** Non-linear. Notice that X values lies between -2 to 2, Y values are very steep. This means, small changes in x would also bring about large changes in the value of Y.
- **Value Range :** 0 to 1
- **Uses :** Usually used in output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to be ***1*** if value is greater than **0.5** and ***0*** otherwise.

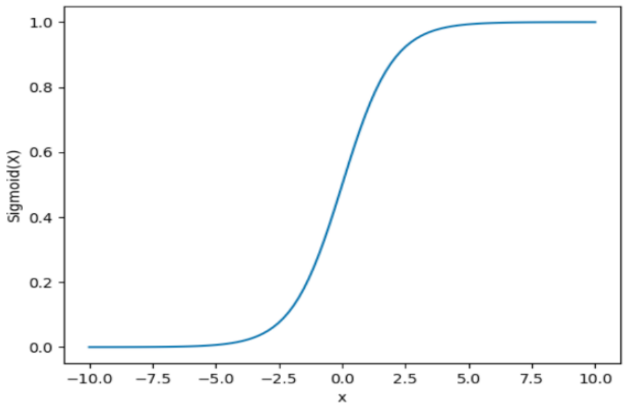


Figure 4.7 Sigmoid Function

3. Tanh Function

- The activation that works almost always better than sigmoid function is Tanh function also known as **Tangent Hyperbolic function**. It's actually mathematically shifted version of the sigmoid function. Both are similar and can be derived from each other **(see in Figure 4.8).**

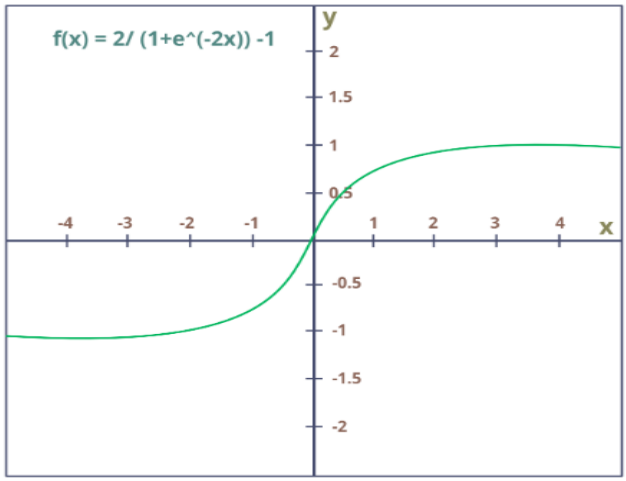


Figure 4.8 Tanh Function

- **Equation :-**

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

- **Value Range :-** -1 to +1
- **Nature :-** non-linear
- **Uses :-** Usually used in hidden layers of a neural network as it's values lies between **-1 to 1** hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in *centering the data* by bringing mean close to 0. This makes learning for the next layer much easier.

4. ReLU Function

- It Stands for *Rectified linear unit*. It is the most widely used activation function. Chiefly implemented in *hidden layers* of Neural network.

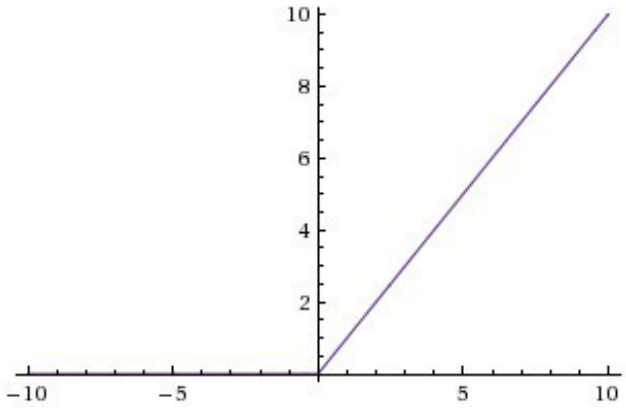


Figure 4.9 ReLU FUNCTION

- **Equation :-** ***A(x) = max(0,x)***. It gives an output x if x is positive and 0 otherwise **(see in Figure 4.9).**
- **Value Range :-** [0, inf]
- **Nature :-** non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.
- **Uses :-** ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.
- In simple words, RELU learns *much faster* than sigmoid and Tanh function.

5. Softmax Function

- It is a subclass of the sigmoid function, the softmax function comes in handy when dealing with multiclass classification issues.
- Used frequently when managing several classes. In the output nodes of image classification issues, the softmax was typically present. The softmax function would split by the sum of the outputs and squeeze all outputs for each category between 0 and 1.

- The output unit of the classifier, where we are actually attempting to obtain the probabilities to determine the class of each input, is where the softmax function is best applied.

4. Discuss in detail about how the network is training.

- A **biological neuron** is composed of multiple **dendrites**, a **nucleus** and a axon. When a stimuli is sent to the brain, it is received through the **synapse** located at the extremity of the dendrite.
- When a **stimuli** arrives at the brain it is transmitted to the neuron via the **synaptic receptors** which **adjust the strength of the signal sent to the nucleus**. This message is **transported** by the **dendrites** to the **nucleus** to then be **processed** in **combination** with other signals emanating from other receptors on the other dendrites. T**hus the combination of all these signals takes place in the nucleus.**
- After processing all these signals, **the nucleus will emit an output signal through its single axon**. The axon will then stream this signal to several other downstream neurons via its **axon terminations**. Thus a neuron analysis is pushed in the subsequent layers of neurons.
- On the other hand, **artificial neural networks** are built on the principle of bio-mimicry. **External stimuli (the data),** whose signal strength is adjusted by the **neuronal weights**, **circulates to the neuron** via the dendrites. The result of the calculation – called the **output** – is then re-transmitted (via the axon) to several other neurons and then subsequent layers are combined, and so on.(see in **Figure 4.10 & 4.11**).

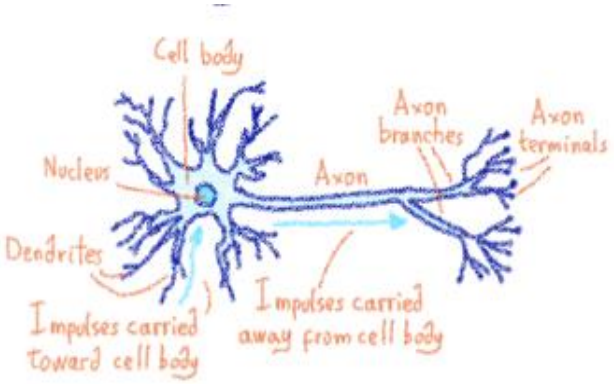


Figure 4.10 Biological Neuron

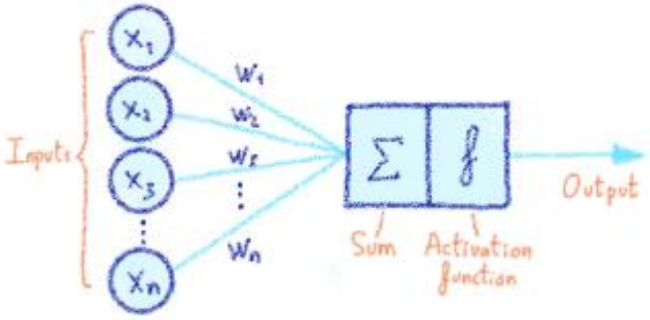


Figure 4.11 Artificial Neuron

Example:

1. Decide on the **number of output classes** (meaning the number of image classes – for example two for cat vs dog)
2. Draw as many computation units as the **number of output classes** (congrats you just create the **Output Layer** of the ANN)
3. Add as many **Hidden Layers** as needed within the defined **architecture**.
4. Stack those **Hidden Layers** to the **Output Layer** using **Neural Connections**
5. It is important to understand that the **Input Layer** is basically a layer of data ingestion
6. Add an **Input Layer** that is adapted to ingest your data
7. Assemble many Artificial Neurons together in a way where the **output** (axon) an **Neuron** on a given **Layer** is (one) of the **input** of another **Neuron** on a subsequent **Layer**. As a consequence, the **Input Layer** is linked to the **Hidden**

Layers which are then linked to the **Output Layer** using **Neural Connections** (also shown in **Figure 4.12**).

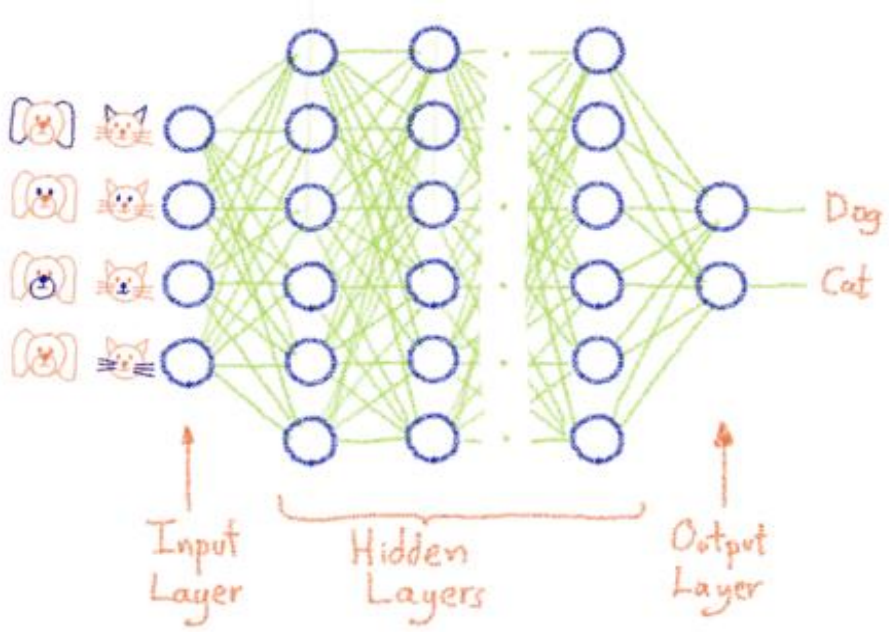


Figure 4.12 A Neural Network Architecture

Train an Artificial Neural Network

- All **Neurons** of a given **Layer** are generating an **Output**, but they don't have the same **Weight** for the next **Neurons Layer**. This means that if a Neuron on a layer observes a given pattern it might mean less for the overall picture and will be partially or completely muted. This is called **Weighting**.
- A **big weight means that the Input is important** and of course **a small weight means that we should ignore it**. Every **Neural Connection** between **Neurons** will have **an associated Weight**.
- **Weights** will be adjusted over the training to fit the **objectives** we have set (recognize that a dog is a dog and that a cat is a cat).
- **In simple terms:** Training a Neural Network means finding the appropriate Weights of the Neural Connections thanks to a feedback loop called Gradient Backward propagation .

Steps to Training an Artificial Neural Network

1. First an ANN will require a **random weight initialization**
2. Split the dataset in **batches (batch size)**
3. Send the batches 1 by 1 to the GPU
4. Calculate the **forward pass** (what would be the output with the current weights)
5. Compare the calculated output to the expected output **(loss)**
6. Adjust the **weights** (using the **learning rate** increment or decrement) according to the **backward pass (backward gradient propagation)**.
7. Go back to step 2

5. Discuss in detail about Gradient descent optimization Algorithm.

- Gradient Descent is a generic optimization algorithm capable of finding optimal solutions to a wide range of problems.
- The general idea is to tweak parameters iteratively in order to minimize the cost function.
- An important parameter of Gradient Descent (GD) is the size of the steps, determined by the learning rate hyperparameters. If the learning rate is too small, then the algorithm will have to go through many iterations to converge, which will take a long time, and if it is too high we may jump the optimal value.

Types of Gradient Descent:

- Typically, there are three types of Gradient Descent:

1. Batch Gradient Descent

- Batch Gradient Descent involves calculations over the full training set at each step as a result of which it is very slow on very large training data. Thus, it becomes very computationally expensive to do Batch GD.

2. Stochastic Gradient Descent

- In SGD, only one training example is used to compute the gradient and update the parameters at each iteration. This can be faster than batch gradient descent but may lead to more noise in the updates.

3. Mini-batch Gradient Descent

- In mini-batch gradient descent, a small batch of training examples is used to compute the gradient and update the parameters at each iteration. This can be a good compromise between batch gradient descent and SGD, as it can be faster than batch gradient descent and less noisy than SGD.

6. Explain in detail about Stochastic gradient descent.

Stochastic Gradient Descent:

- In Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration.
- In Gradient Descent, there is a term called "batch" which denotes the total number of samples from a dataset that is used for calculating the gradient for each iteration.
- In typical Gradient Descent optimization, like Batch Gradient Descent, the batch is taken to be the whole dataset. Although using the whole dataset is really useful for getting to the minima in a less noisy and less random manner, the problem arises when our dataset gets big.
- Suppose, you have a million samples in your dataset, so if you use a typical Gradient Descent optimization technique, you will have to use all of the one million samples for completing one iteration while performing the Gradient Descent, and it has to be done for every iteration until the minima are reached. Hence, it becomes computationally very expensive to perform.
- This problem is solved by Stochastic Gradient Descent. In SGD, it uses only a single sample, i.e., a batch size of one, to perform each iteration.
- The sample is randomly shuffled and selected for performing the iteration.

SCD Algorithm

$$\text{for } i \text{ in range } (m):$$
$$\theta_j = \theta_j - \alpha \left(\widehat{y}^i - y^i\right) x_j^i$$

- In SGD, we find out the gradient of the cost function of a single example at each iteration instead of the sum of the gradient of the cost function of all the examples.
- In SGD, since only one sample from the dataset is chosen at random for each iteration, the path taken by the algorithm to reach the minima is usually noisier than your typical Gradient Descent algorithm. But that doesn't matter all that much because the path taken by the algorithm does not matter, as long as we reach the minima and with a significantly shorter training time **(see in Figure 4.13 & 4.14)**.

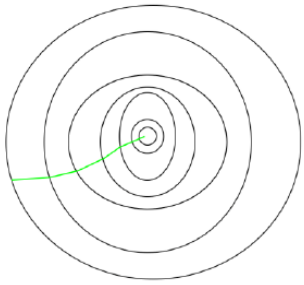


Figure 4.13 The path taken by Batch Gradient Descent

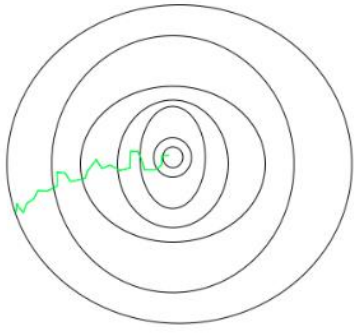


Figure 4.14 The path taken by Stochastic Gradient Descent

- SGD is generally noisier than typical Gradient Descent, it usually took a higher number of iterations to reach the minima, because of its randomness in its descent.
- Even though it requires a higher number of iterations to reach the minima than typical Gradient Descent, it is still computationally much less expensive than typical Gradient Descent. Hence, in most scenarios, SGD is preferred over Batch Gradient Descent for optimizing a learning algorithm.

Advantages:

- **Speed:** SGD is faster than other variants of Gradient Descent.
- **Memory Efficiency:**it is memory-efficient and can handle large datasets that cannot fit into memory.
- **Avoidance of Local Minima:** Due to the noisy updates in SGD, it has the ability to escape from local minima and converge to a global minimum.

Disadvantages:

- **Noisy updates:** The updates in SGD are noisy and have a high variance, which can make the optimization process less stable and lead to oscillations around the minimum.
- **Slow Convergence:** SGD may require more iterations to converge to the minimum since it updates the parameters for each training example one at a time.
- **Sensitivity to Learning Rate:** The choice of learning rate can be critical in SGD since using a high learning rate can cause the algorithm to overshoot the minimum, while a low learning rate can make the algorithm converge slowly.
- **Less Accurate:** Due to the noisy updates, SGD may not converge to the exact global minimum and can result in a suboptimal solution. This can be mitigated by using techniques such as learning rate scheduling and momentum-based updates.

7. Explain in detail about error backpropagation.

Explain about the back propagation algorithm in neural network. [Nov 2024]

- **Backpropagation** is one of the important concepts of a neural network. or a single training example, **Backpropagation** algorithm calculates the gradient of the **error function**.
- Backpropagation algorithms are a set of methods used to efficiently train artificial neural networks following a gradient descent approach which exploits the chain rule.
- The **main features of Backpropagation are the iterative**, recursive and efficient method through which it calculates the updated weight to improve the network until it is not able to perform the task for which it is being trained.
- Derivatives of the activation function to be known at network design time is required to Backpropagation.

How Backpropagation Algorithm Works?

- The Back propagation algorithm in neural network computes the gradient of the loss function for a single weight by the chain rule. It efficiently computes one layer at a time, unlike a native direct computation. It computes the gradient, but it does not define how the gradient is used. It generalizes the computation in the delta rule.(see in **Figure 4.16**)

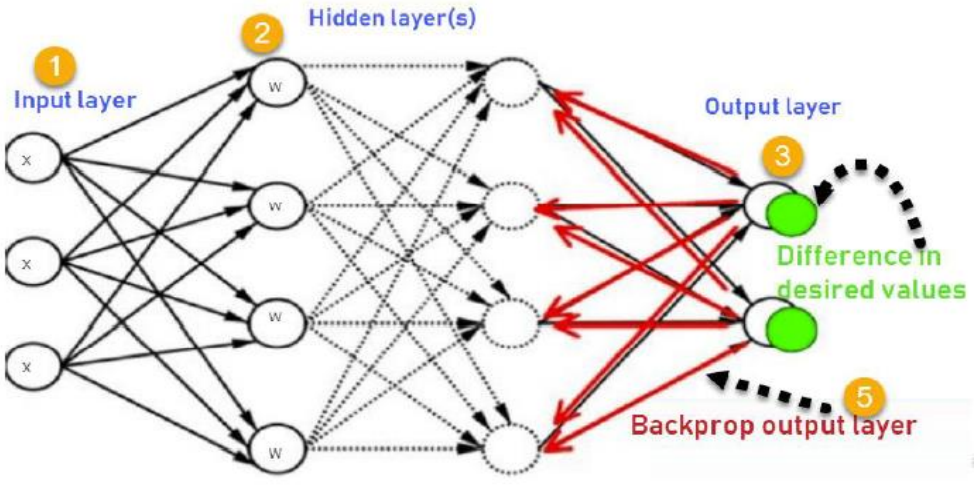


Figure 4.16 Back propagation neural network

1. Inputs X, arrive through the preconnected path
2. Input is modeled using real weights W. The weights are usually randomly selected.
3. Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
4. Calculate the error in the outputs

$Error_B$= Actual Output – Desired Output

5. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.
6. Keep repeating the process until the desired output is achieved

Types of Backpropagation Networks

- Two Types of Backpropagation Networks are:
    a. Static Back-propagation
    b. Recurrent Backpropagation

Static back-propagation:

- It is one kind of backpropagation network which produces a mapping of a static input for static output. It is useful to solve static classification issues like optical character recognition.

Recurrent Backpropagation:

- Recurrent Back propagation in data mining is fed forward until a fixed value is achieved. After that, the error is computed and propagated backward.

Advantages:

- It does not have any parameters to tune except for the number of inputs.
- It is highly adaptable and efficient and does not require any prior knowledge about the network.
- It is a standard process that usually works well.

Disadvantages:

- The performance of backpropagation relies very heavily on the training data.
- Backpropagation needs a very large amount of time for training.
- Backpropagation requires a matrix-based method instead of mini-batch.

8. Explain in detail about Unit saturation (aka the vanishing gradient problem).

- The vanishing gradient problem is an issue that sometimes arises when training machine learning algorithms through gradient descent. This most often occurs in neural networks that have several neuronal layers such as in a deep learning system, but also occurs in recurrent neural networks.
- The key point is that the calculated partial derivatives used to compute the gradient as one goes deeper into the network. Since the gradients control how much the network learns during training, the gradients are very small or zero, then little to no training can take place, leading to poor predictive performance.

The problem:

- As more layers using certain activation functions are added to neural networks, the gradients of the loss function approaches zero, making the network hard to train.

Why:

- Certain activation functions, like the sigmoid function, squishes a large input space into a small input space between 0 and 1. Therefore, a large change in the input of the sigmoid function will cause a small change in the output. Hence, the derivative becomes small.

The sigmoid function and its derivative

- As an example, the below **Figure 4.17** is the sigmoid function and its derivative. Note how when the inputs of the sigmoid function becomes larger or smaller (when $|x|$ becomes bigger), the derivative becomes close to zero.

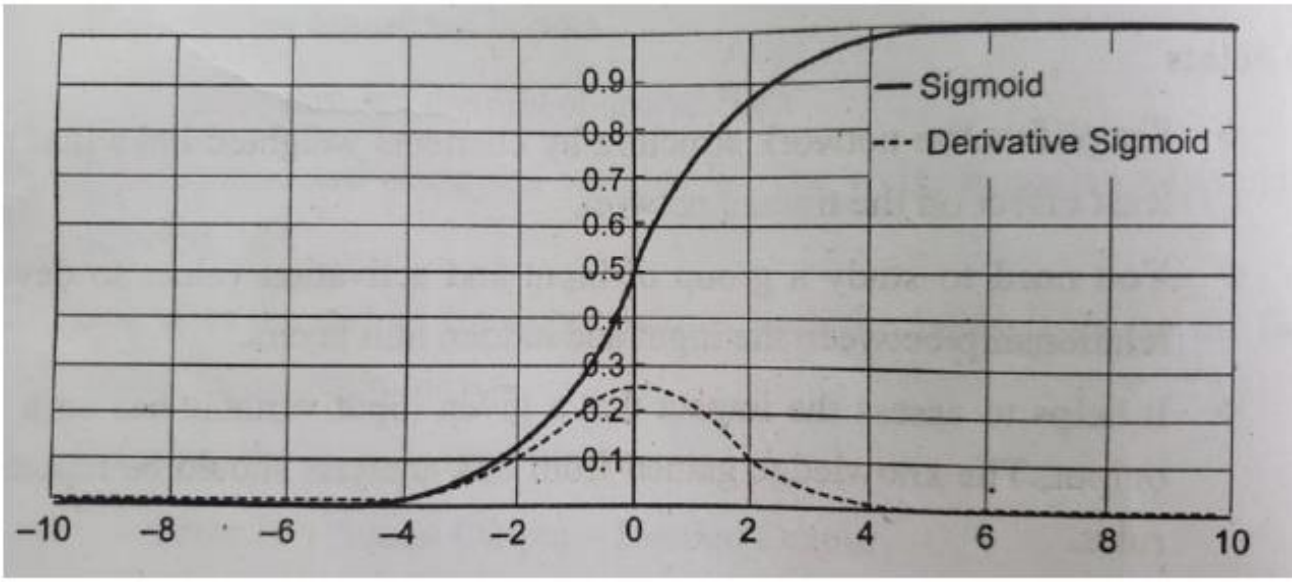


Figure 4.17 The sigmoid function and its derivative

Why it's significant:

- For shallow network with only a few layers that use these activations, this isn't a big problem. However, when more layers are used, it can cause the gradient to be too small for training to work effectively. Gradients of neural networks are found using backpropagation. Simply put, backpropagation finds the derivatives of the network by moving layer by layer from the final layer to the initial one. By the chain rule, the derivatives of each layer are multiplied down the network (from the final layer to the initial) to compute the derivatives of the initial layers.
- However, when n hidden layers use activation like the sigmoid an function, n small derivatives are multiplied together. Thus, the gradient decreases exponentially as we propagate down to the initial layers. A small gradient means that the weights and biases of the initial layers will not be updated effectively with each training session. Since these initial layers are often crucial to recognizing the core elements of the input data, it can lead to overall inaccuracy of the whole network.

Solution:

- The simplest solution is to use other activation functions, such as ReLU, which doesn't cause a small derivative. Residual networks are another solution, as they provide residual connections straight to earlier layers.
- The residual connection directly adds the value at the beginning of the block, x, to the end of the block (F(x) + x). This residual connection doesn't go through activation functions that "squashes" the derivatives, resulting in a higher overall derivative of the block.(see in **Figure 4.18)**

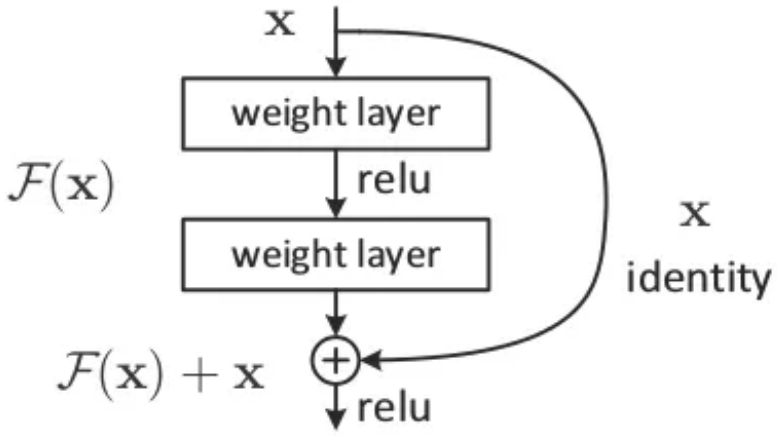


Figure 4.18 A residual block

- Finally, batch normalization layers can also resolve the issue. As stated before, the problem arises when a large input space is mapped to a small one, causing the derivatives to disappear.

9. Explain in detail about Rectified linear unit (ReLU).

- It Stands for *Rectified linear unit*. It is the most widely used activation function. Chiefly implemented in *hidden layers* of Neural network. (see the below **figure 4.19**)

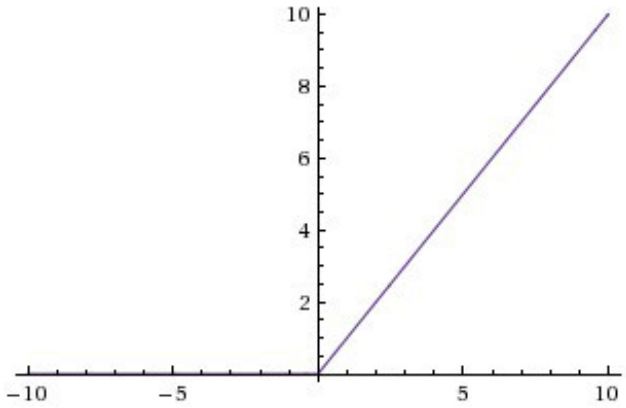


Figure 4.19 RELU FUNCTION

- **Equation :-** ***A(x) = max(0,x)***. It gives an output x if x is positive and 0 otherwise.
- **Value Range :-** [0, inf)
- **Nature :-** non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.
- **Uses :-** ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.
- In simple words, RELU learns *much faster* than sigmoid and Tanh function.
- An activation function for hidden units that has become popular recently with deep networks is the *rectified linear unit* (ReLU), which is defined as

$$\text{ReLU}(a) = \begin{cases} a & \text{if } a > 0 \\ 0 & \text{otherwise} \end{cases}$$

Though ReLU is not differentiable at $a = 0$, we use it anyway; we use the left derivative:

$$\text{ReLU}'(a) = \begin{cases} 1 & \text{if } a > 0 \\ 0 & \text{otherwise} \end{cases}$$

Leaky ReLU

- In the *leaky ReLU* (the output is also linear on the negative side but with a smaller slope, just enough to make sure that there will be updates for negative activations, albeit small:

$$\text{leaky-ReLU}(a) = \begin{cases} a & \text{if } a > 0 \\ \alpha a & \text{otherwise} \end{cases}$$

Advantage:

- it does not saturate (unlike sigmoid and tanh), updates can still be done for large positive a for some inputs, some hidden unit activations will be zero, meaning that we will have a sparse representation
- Sparse representations lead to faster Training

Disadvantage:

- The derivative is zero for $a \leq 0$, there is no further training if, for a hidden unit, the weighted sum somehow becomes negative. This implies that one should be careful in initializing the weights so that the initial activation for all hidden units is positive.

10. Explain in detail about Batch Normalization.

Illustrate the following

(i) Batch Normalization

(ii) Dropout [Apr 2024]

Normalization

- Normalization is a data pre-processing tool used to bring the numerical data to a common scale without distorting its shape.

Batch normalization

- **Batch normalization** is a process to make neural networks faster and more stable through adding extra layers in a deep neural network. The new layer

performs the standardizing and normalizing operations on the input of a layer coming from a previous layer. A typical neural network is trained using a collected set of input data called **batch**. Similarly, the normalizing process in batch normalization takes place in batches, not as a single input.

- A similar case can also be made for the hidden units, and this is the idea behind *batch normalization.*
- We normalize hidden unit values *before* applying the activation function, such as sigmoid or ReLU. Let us call that weighted sum *aj*. For each batch or minibatch, for each hidden unit *j* we calculate the mean *mj* and standard deviation *sj* of its values, and we first *z*-normalize:

$$\tilde{a}_j = \frac{a_j - m_j}{s_j}$$

- We can then map these to have arbitrary mean γj and scale βj and then we apply the activation function.

$$\hat{a}_j = \gamma_j \tilde{a}_j + \beta_j$$

- First, *mj* and *sj* are calculated anew for each batch, and we see immediately that batch normalization is not meaningful with online learning or very small minibatches.
- Second, γj and βj are parameters that are initialized and updated (after each batch or minibatch) using gradient descent, just like the connection weights. So they require extra memory and computation.
- These new parameters allow each hidden unit to have its arbitrary (but learned) mean and standard deviation for its activation.
- Once learning is done, we can go over the whole training data (or a large enough subset) and calculate *mj* and *sj* to use during testing, when we see one instance at a time.

Why batch normalization?

- **An internal covariate shift** occurs when there is a change in the input distribution to our network. When the input distribution changes, hidden layers try to learn to adapt to the new distribution.
- This slows down the training process. If a process slows down, it takes a long time to converge to a global minimum.

Example:

- Suppose we are training an image classification model, that classifies the images into Dog or Not Dog. Let's say we have the images of white dogs only, these images will have certain distribution as well. Using these images model will update its parameters.
- Later, if we get a new set of images, consisting of non-white dogs. These new images will have a slightly different distribution from the previous images. Now the model will change its parameters according to these new images.
- Hence the distribution of the hidden activation will also change. This change in hidden activation is known as an internal covariate shift. Here batch normalization works.

Advantages of Batch Normalization

- Speed Up the Training
- Handles internal covariate shift
- The model is less delicate to hyperparameter tuning**.**
- Batch normalization smoothens the loss function that in turn by optimizing the model parameters improves the training speed of the model.

11. Explain in detail about hyperparameter tuning.

- A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. By training a model with existing data, we are able to fit the model parameters.

- However, there is another kind of parameter, known as ***Hyperparameters***, that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.
- Some examples of model hyperparameters include:
    1. The penalty in Logistic Regression Classifier i.e. L1 or L2 regularization
    2. The learning rate for training a neural network.
    3. The C and sigma hyperparameters for support vector machines.
    4. The k in k-nearest neighbors.
- Models can have many hyperparameters and finding the best combination of parameters can be treated as a search problem. The two best strategies for Hyperparameter tuning are:
    1. GridSearchCV
    2. RandomizedSearchCV

GridSearchCV

- In GridSearchCV approach, the machine learning model is evaluated for a range of hyperparameter values. This approach is called GridSearchCV, because it searches for the best set of hyperparameters from a grid of hyperparameters values.
- **For example,** if we want to set two hyperparameters C and Alpha of the Logistic Regression Classifier model, with different sets of values. The grid search technique will construct many versions of the model with all possible combinations of hyperparameters and will return the best one.
- As in the image, for C = [0.1, 0.2, 0.3, 0.4, 0.5] and Alpha = [0.1, 0.2, 0.3, 0.4]. For a combination of ***C=0.3 and Alpha=0.2***, the performance score comes out to be **0.726(Highest)**, therefore it is selected. (see in **Figure 5.20)**

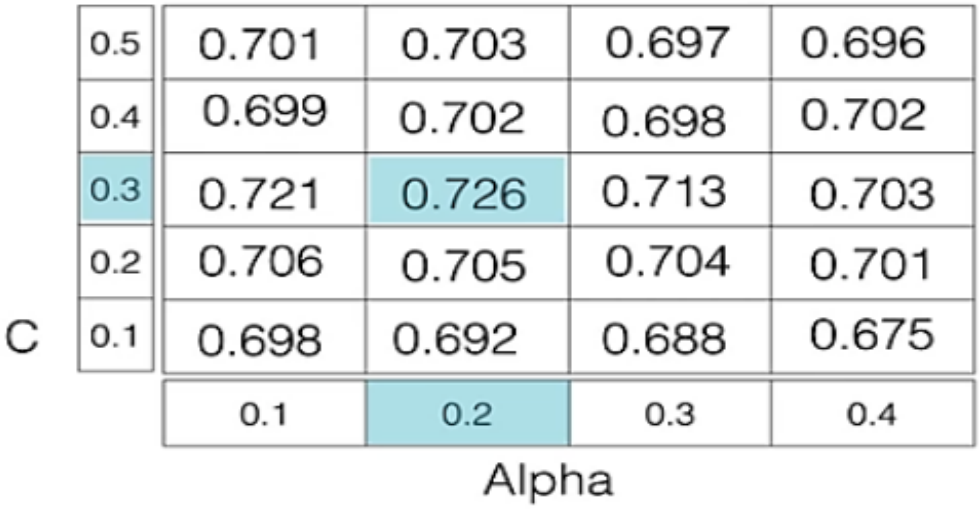


Figure 5.20 GridSearchCV

The following code illustrates how to use GridSearchCV

 # Necessary imports

from sklearn.linear_model import Logistic Regression

from sklearn.model_selection import GridSearchCV

 # Creating the hyperparameter grid

c_space = np.logspace(-5, 8, 15) param_grid = {'C': c_space}

#Instantiating logistic regression classifier

logreg = Logistic Regression()

 # Instantiating the GridSearchCV object

logreg_cv= GridSearchCV(logreg, param_grid, cv = 5)

logreg_cv.fit(X,y)

 # Print the tuned parameters and score

print("Tuned Logistic Regression Parameters:{}".format(logreg_cv.best_params_))

print("Best score is {}".format(logrcg_cv.best_score_))

Output:

Tuned Logistic Regression Parameters: {'C': 3.7275937203149381) Best score is 0.7708333333333334

Drawback:

GridSearch CV will go through all the intermediate combinations of hyperparameters which makes grid search computationally very expensive.

RandomizedSearchCV

- RandomizedSearchCV solves the drawbacks of GridSearchCV, as it goes through only a fixed number of hyperparameter settings.
- It moves within the grid in a random fashion to find the best set of hyperparameters. This approach reduces unnecessary computation.

12. Explain in detail about Regularization.

Overfitting:

- When a model performs well on the training data and does not perform well on the testing data, then the model is said to have high generalization error. In other words, in such a scenario, the model has low bias and high variance and is too complex. This is called overfitting.
- Overfitting means that the model is a good fit on the train data compared to the data. Overfitting is also a result of the model being too complex

Regularization:

- Regularization is one of the most important concepts of machine learning. It is a technique to prevent the model from overfitting by adding extra information to it. Regularization helps choose a simple model rather than a complex one.
- Generalization error is "a measure of how accurately an algorithm can predict outcome values for previously unseen data." Regularization refers to the modifications that can be made to a leaming algorithm that helps to reduce this generalization error and not the training error.

The commonly used regularization techniques are:

1. Hints

- *Hints* are properties of the target function that are known to us independent of the training examples

Figure 4.21 HINTS

- The identity of the object does not change when it is translated, rotated, or scaled. Note that this may not always be true, or may be true up to a point: 'b' and 'q' are rotated versions of each other. These are hints that can be incorporated into the learning process to make learning easier.
- In image recognition, there are invariance hints: The identity of an object does not change when it is rotated, translated, or scaled (see **Figure 4.21).** Hints are auxiliary information that can be used to guide the learning process and are especially useful when the training set is limited.
- There are different ways in which hints can be used:
    - Hints can be used to create *virtual examples*.
    - The hint may be incorporated into the network structure.

2. Weight decay:

- Incentivize the network to use smaller weights by adding a penalty to the loss function.
- Even if we start with a weight close to zero, because of some noisy instances, it may move away from zero; the idea in *weight decay* is to add some small constant background force that always pulls a weight toward zero, unless it is absolutely necessary that it be large (in magnitude) to decrease error. For any weight *wi*, the update rule is

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} - \lambda' w_i$$

3. Ridge Regression

- The Ridge regression technique is used to analyze the model where the variables may be having multicollinearity.
- It reduces the insignificant independent variables though it does not remove them completely. This type of regularization uses the L_2 norm for regularization.

$$E' = E + \frac{\lambda}{2} \sum_i w_i^2$$

4. Lasso Regression

- Least Absolute Shrinkage and Selection Operator (or LASSO) Regression penalizes the coefficients to the extent that it becomes zero.
- It eliminates the insignificant independent variables. This regularization technique uses the L1 norm for regularization.

$$E' = E + \frac{\lambda}{2} \sum_i |w_i|$$

Illustrate the following **[Apr 2024]**

(ii) Dropout

5. Dropout

- "**Dropout**" in machine learning refers to the process of randomly ignoring certain nodes in a layer during training.
- In the **Figure 4.22**, the neural network on the left represents a typical neural network where all units are activated. On the right, the red units have been

dropped out of the model- the values of their weights and biases are not considered during training.

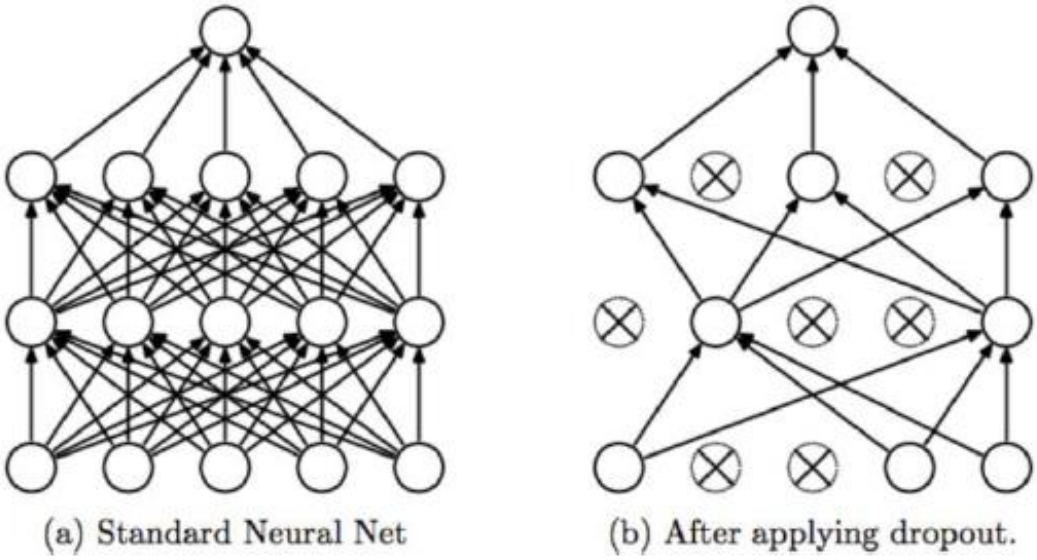


Figure 4.22 Dropout

- Dropout is used as a regularization technique - it prevents overfitting by ensuring that no units are codependent.
- In *dropout*, we have a hyperparameter *p*, and we drop the input or hidden unit with probability *p*, that is, set its output to zero, or keep it with probability 1 – *p*.
- *p* is adjusted using cross-validation; with more inputs or hidden units in a layer, we can afford a larger *p* (see **Figure 4.23**).

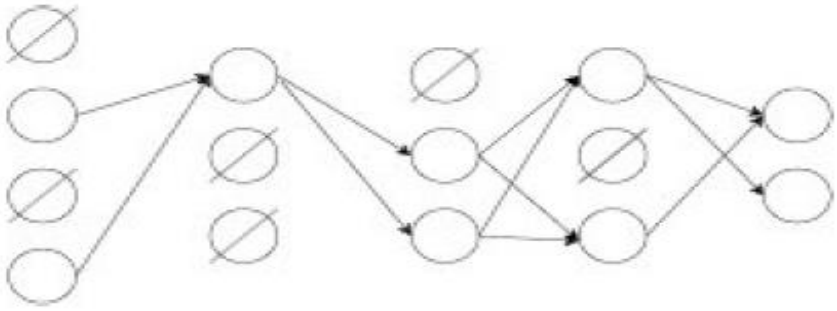
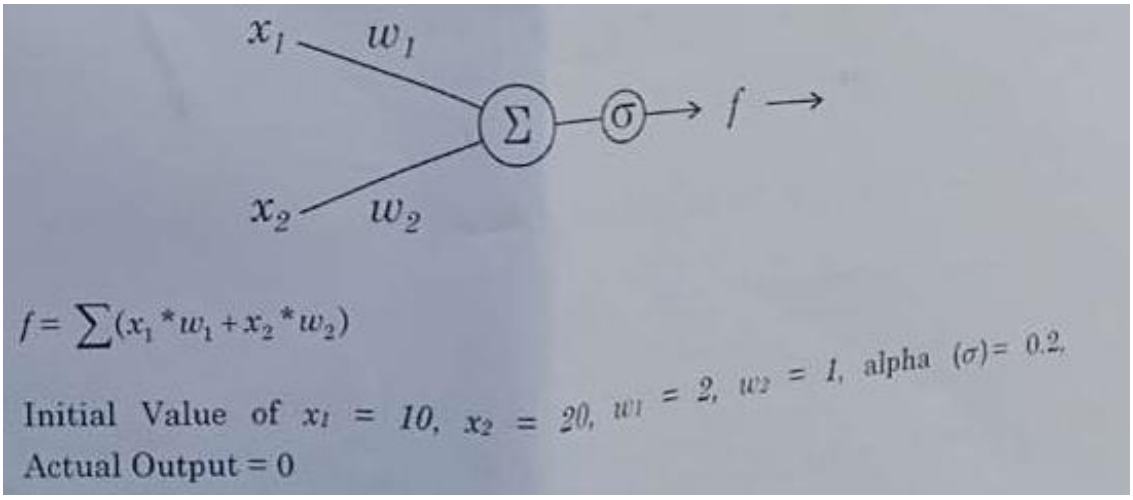


Figure 4.23 cross validation

- In dropout, the output of a random subset of the units are set to zero, and backpropagation is done on the remaining smaller network.
- In each batch or minibatch, for each unit independently we decide randomly to keep it or not. Let us say that p = 0.25. So, on average, we remove a quarter of the units and we do backpropagation as usual on the remaining network for that batch or minibatch. We need to make up for the loss of units, though: In each layer, we divide the activation of the remaining units by $1 - p$ to make sure that they provide a vector of similar magnitude to the next layer. There is no dropout during testing.
- In each batch or minibatch, a smaller network (with smaller variance) is trained. Thus dropout is effectively sampling from a pool of possible networks of different depths and widths.
- There is a version called *dropconnect* that drops out or not connections independently, which allows a larger set of possible networks to sample from, and this may be preferable in smaller networks.

13. Consider the following neural network problem, use back propagation algorithm to find the update of w1.



Use mean square error as loss function. **[Nov 2024]**

Step 1: Define the Forward Pass

The given function is:

$$F = x_1 w_1 + x_2 w_2$$

Given values:

$$x_1 = 10, \quad x_2 = 20, \quad w_1 = 2, \quad w_2 = 1$$

Compute the output:

$$F = (10 \times 2) + (20 \times 1) = 20 + 20 = 40$$

Step 2: Compute the Loss using Mean Squared Error (MSE)

The actual output is given as:

$$y_{\text{actual}} = 0$$

MSE Loss function:

$$L = \frac{1}{2}(F - y_{\text{actual}})^2$$

$$L = \frac{1}{2}(40 - 0)^2 = \frac{1}{2}(1600) = 800$$

Step 3: Compute the Gradient (Backpropagation)

We need to compute:

$$\frac{\partial L}{\partial w_1}$$

Using the chain rule:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial F} \times \frac{\partial F}{\partial w_1}$$

First, compute:

$$\frac{\partial L}{\partial F} = (F - y_{\text{actual}}) = (40 - 0) = 40$$

Next,

$$\frac{\partial F}{\partial w_1} = x_1 = 10$$

So,

$$\frac{\partial L}{\partial w_1} = 40 \times 10 = 400$$

Step 4: Update w_1 Using Gradient Descent

Gradient Descent update rule:

$$w_1^{\text{new}} = w_1 - \alpha \frac{\partial L}{\partial w_1}$$

Given learning rate:

$$\alpha = 0.2$$

$$w_1^{\text{new}} = 2 - (0.2 \times 400)$$

$$w_1^{\text{new}} = 2 - 80$$

$$w_1^{\text{new}} = -78$$

Final Answer: Updated w_1

$$w_1 = -78$$

14. Write about notes on Hyper parameter tunning. [Nov 2024]

- Hyperparameter tuning is the process of selecting the best hyperparameters for a machine learning model to improve its performance.
- Unlike model parameters (which are learned from the data), hyperparameters are set before training and control the learning process.

1. Types of Hyperparameters

- Hyperparameters can be categorized into two main types:

a) Model-Specific Hyperparameters

These define the structure of the model. Examples:

- **Number of hidden layers** (in neural networks)
- **Number of neurons per layer**
- **Kernel size** (in CNNs)
- **Tree depth** (in decision trees)

b) Training-Specific Hyperparameters

These control the learning process. Examples:

- **Learning rate (α):** Controls step size in gradient descent
- **Batch size:** Number of samples processed at once
- **Number of epochs:** How many times the model sees the entire dataset
- **Regularization parameter (λ):** Prevents overfitting (L1, L2 regularization)

2. Methods for Hyperparameter Tuning

Several techniques help find the best hyperparameters:

a) Manual Search

- Adjust hyperparameters based on intuition and past experience.
- Simple but time-consuming and not always effective.

b) Grid Search

- Tries all possible combinations of hyperparameters.
- Works well for small parameter spaces but is computationally expensive.

c) Random Search

- Selects random combinations of hyperparameters.
- More efficient than grid search for large parameter spaces.

d) Bayesian Optimization

- Uses probabilistic models to find the best hyperparameters efficiently.
- Balances exploration (trying new values) and exploitation (refining known good values).

e) Genetic Algorithms & Evolutionary Strategies

- Inspired by biological evolution (mutation, crossover).
- Helps in exploring complex hyperparameter spaces.

3. Common Challenges in Hyperparameter Tuning

1. **Overfitting:**
    - Using too many parameters can lead to overfitting.
    - Solutions: Cross-validation, dropout, regularization.
2. **Underfitting:**
    - Model too simple to learn patterns.
    - Solution: Increase model complexity.
3. **Computational Cost:**
    - Tuning can be expensive.
    - Solution: Use random search or Bayesian optimization.
4. **Interaction Between Hyperparameters:**
    - Some hyperparameters depend on others.
    - Example: Learning rate and batch size impact each other.

4. Best Practices for Hyperparameter Tuning

- **Start with default values** and adjust systematically.
- **Use smaller subsets** of data for faster testing.
- **Use cross-validation** to validate performance.
- **Leverage automation tools** like **Optuna, Hyperopt, and Ray Tune**.
- **Monitor results** using visualizations like learning curves.

Hyperparameter tuning is essential for building efficient and accurate models. Choosing the right approach depends on available resources, dataset size, and computational power.

UNIT V DESIGN AND ANALYSIS OF MACHINE LEARNING EXPERIMENTS

Guidelines for machine learning experiments, Cross Validation (CV) and resampling – K-fold CV, bootstrapping, measuring classifier performance, assessing a single classification algorithm and comparing two classification algorithms – t test, McNemar's test, K-fold CV paired t test.

PART-A

1. Recall the benefits of the cross validation method. [Apr 2024]

- Reduces Overfitting: Helps ensure the model generalizes well to unseen data.
- Improves Model Performance Estimation: Provides a more reliable measure of model accuracy.
- Utilizes Data Efficiently: Maximizes the use of available data by using different training and testing splits.
- Works Well with Small Datasets: Ensures robust evaluation even when data is limited.
- Provides More Reliable Metrics: Reduces variance in performance evaluation compared to a single train-test split.
- Helps in Hyperparameter Tuning: Assists in selecting the best hyperparameters by testing multiple configurations.
- Cross-validation is essential for building robust and generalizable machine learning models.

2. How do you evaluate a classification algorithm? [Apr 2024]

- Evaluating a classification algorithm involves assessing its performance using various metrics and techniques.

1. Use a Proper Data Split
    - Train-Test Split: Divide the dataset into training and testing sets (e.g., 80% train, 20% test).
    - Cross-Validation: Use k-fold cross-validation to get a more reliable estimate of model performance.
2. Performance Metrics
    - Key evaluation metrics for classification models:

a) Accuracy

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

- Useful when classes are balanced.
- Can be misleading if classes are imbalanced.
- Precision, Recall, and F1-Score

Precision: Measures how many of the predicted positives are actually positive.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{TP} + \text{False Positives (FP)}}$$

- **Recall (Sensitivity)**: Measures how many actual positives were correctly identified.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{False Negatives (FN)}}$$

- **F1-Score**: Harmonic mean of precision and recall (useful for imbalanced datasets).

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

3. Write the advantages of cross validation. [Nov 2024]

- **Better Model Performance Estimation:** Provides a more reliable assessment of how the model will perform on unseen data.
- **Reduces Overfitting:** Ensures the model generalizes well by testing it on multiple subsets of data.
- **Maximizes Data Usage:** Uses the entire dataset for training and testing, improving efficiency, especially for small datasets.
- **Reduces Variance in Results:** Provides a more stable estimate of model performance compared to a single train-test split.
- **Helps in Hyperparameter Tuning:** Aids in selecting the best hyperparameters by testing different configurations.
- **Works Well with Imbalanced Data:** Ensures each class gets represented in training and validation sets properly.
- **Provides Robust Comparisons:** Helps compare different models fairly by evaluating them on multiple data splits.

4. Define the concept of bootstrapping. [Nov 2024]

- Bootstrapping is a resampling technique used to estimate the distribution of a statistic (e.g., mean, variance, accuracy) by repeatedly sampling with replacement from a given dataset.
- It is widely used in statistics and machine learning for estimating confidence intervals, hypothesis testing, and model validation.

5. What are the Guidelines for machine learning experiments?

- Data Collection and integration
- Exploratory Data Analysis and Visualisation
- Feature Selection and Engineering
- Model Training
- Model Evaluation
- Prediction

6. What is resampling?

- In machine learning, resampling is a method that involves repeatedly drawing samples from the training dataset. The samples are then used to refit a specific model to retrieve more information about the fitted model.

7. What is Cross Validation (CV)?

- Cross-validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. We can also say that it is a technique to check how a statistical model generalizes to an independent dataset.

8. What are the basic steps of cross-validations?

- Reserve a subset of the dataset as a validation set.
- Provide the training to the model using the training dataset.
- Now, evaluate model performance using the validation set. If the model performs well with the validation set, perform the further step, else check for the issues.

9. What are the Methods used for Cross-Validation?

There are some common methods that are used for cross-validation. These methods are given below:

1. Validation Set Approach
2. Leave-P-out cross-validation
3. Leave one out cross-validation
4. K-fold cross-validation
5. Stratified k-fold cross-validation

10. What is K-fold CV?

- K-fold cross-validation approach divides the input dataset into K groups of samples of equal sizes. These samples are called **folds**. For each learning set, the prediction function uses k-1 folds, and the rest of the folds are used for the test set. This approach is a very popular CV approach because it is easy to understand, and the output is less biased than other methods.

11. Define bootstrapping.(Nov/Dem 2024)

- Bootstrapping is a resampling technique in machine learning that involves repeatedly drawing samples from source data with replacement. The term comes from the idea of lifting oneself up without external help by pulling on one's own bootstraps.

12. List and explain the types of bootstrapping methods.

1. **Parametric Bootstrap Method**
   - In this method, the distribution parameter must be known. This means that the assumption of the kind of distribution the sample has must be provided beforehand.
2. **Non-Parametric Bootstrap Method**
   - Unlike the parametric bootstrap method, this type does not require the parameter of distribution to be known beforehand. Therefore, this type of bootstrap method works without assuming the nature of the sample distribution.

13. What is Measuring classifier performance?

- Evaluating the performance of a Machine learning model is one of the important steps while building an effective ML model.
- *To evaluate the performance or quality of the model, different metrics are used, and these metrics are known as performance metrics or evaluation metrics.*

14. Recall the benefit of cross - validation method. (Apr/May , Nov/Dem 2024)

- Cross-validation has several advantages over other methods of model evaluation. It reduces the risk of overfitting, by testing the model on different subsets of the data.
- This ensures that the model can perform well on various samples, not just on one specific split.

15. How do you evaluate the classification algorithm. (Nov/Dem2024)

- to Evaluate Classification Models
- Precision (Positive Predicted Value)
- Recall (Sensitivity, True Positive Rate)
- Specificity (Selectivity, True Negative Rate)
- Fall-out (False Positive Rate)
- Miss Rate (False Negative Rate)
- Receiver-Operator Curve (ROC Curve) and Area Under the Curve (AUC)

16. What is T-Test?

- A **statistical hypothesis test** used to compare the means of two groups and determine if they are significantly different.
- Common types: **Independent t-test** (for two independent samples) and **Paired t-test** (for related samples, like before-and-after measurements).

17. What is McNemar's Test?

- A **non-parametric test** used to compare two related categorical datasets, often in classification tasks.
- Used when the same subjects are tested twice (e.g., before and after applying a new model) to check if the changes are statistically significant.

18. What is K-Fold Cross-Validation Paired T-Test?

- A technique to evaluate model performance by splitting the dataset into **K folds**, training on **K-1** folds, and testing on the remaining one.
- A **paired t-test** can then be used to compare the performance of two models across different folds to check if one model significantly outperforms the other.

19. How to Measuring Classifier Performance?

- Common metrics: Accuracy, Precision, Recall, F1-score, ROC-AUC.
- Performance can be evaluated using K-fold CV, bootstrapping, or holdout validation methods.

20. Assessing a Single Classification Algorithm (2 Marks)

- Evaluate model performance using metrics like Accuracy, Precision, Recall, F1-score, and AUC-ROC.
- Use Cross-Validation (e.g., K-Fold CV) to ensure reliability.
- Consider confusion matrix analysis to understand false positives and false negatives.

21. Comparing Two Classification Algorithms (2 Marks)

- Use paired statistical tests like the Paired t-test or McNemar's test to check if one model is significantly better.
- Compare performance using K-Fold CV across multiple splits.
- Use performance metrics (e.g., Precision-Recall curves, AUC-ROC) to analyze differences.
- If applicable, compare models in terms of computational efficiency and interpretability.

PART-B

1. What are the Guidelines for machine learning experiments?

Guidelines for machine learning experiments

The Guidelines for machine learning experiments are shown in figure 5.1.

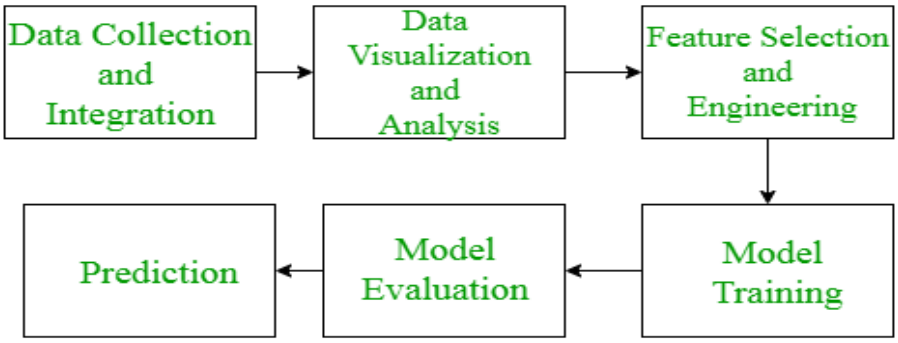


Figure 5.1 Guidelines for machine learning experiments

1. Data Collection and integration:

- Data collected acts as an input to the model (data preparation phase).
- Inputs are called features.
- Data collected in the case of our considered example involves a lot of data. For example, the collected data should answer the following questions- What is past customer history? What were the past orders? Is the customer a prime member of our bookstore? Does the customer own a kindle? Has the customer made any previous complaints? What was the most number of complaints?
- The more the data is, more the better our model becomes.
- Once the data is collected we need to integrate and prepare the data.
- Integration of data means placing all related data together.
- Then data preparation phase starts in which we manually and critically explore the data.
- The data preparation phase tells the developer that is the data matching the expectations.

2. Exploratory Data Analysis and Visualisation:

- Once the data is prepared developer needs to visualize the data to have a better understanding of relationships within the dataset.
- When we get to see data, we can notice the unseen patterns that we may not have noticed in the first phase.
- It helps developers easily identify missing data and outliers.
- Data visualization can be done by plotting histograms, scatter plots, etc.

- After visualization is done data is analyzed so that developer can decide what ML technique he may use.
- In the considered example case unsupervised learning may be used to analyze customer purchasing habits.

3. Feature Selection and Engineering:

- Feature selection means selecting what features the developer wants to use within the model.
- Features should be selected so that a minimum correlation exists between them and a maximum correlation exists between the selected features and output.
- Feature engineering is the process to manipulate the original data into new and potential data that has a lot many features within it.
- In simple words Feature engineering is converting raw data into useful data or getting the maximum out of the original data.
- Feature engineering is arguably the most crucial and time-consuming step of the ML pipeline.

4. Model Training:

- After the first three steps are done completely we enter the model training phase.
- It is the first step officially when the developer gets to train the model on basis of data.
- To train the model, data is split into three parts- Training data, validation data, and test data.
- Around 70%-80% of data goes into the training data set which is used in training the model.
- Validation data is also known as development set or dev set and is used to avoid overfitting or underfitting situations i.e. enabling hyperparameter tuning.
- Hyperparameter tuning is a technique used to combat overfitting and underfitting.
- Validation data is used during model evaluation.
- Around 10%-15% of data is used as validation data.
- Rest 10%-15% of data goes into the test data set. Test data set is used for testing after the model preparation.

- It is crucial to randomize data sets while splitting the data to get an accurate model.
- Data can be randomized using Scikit learn in python.

5. Model Evaluation:

- After the model training, validation, or development data is used to evaluate the model.
- To get the most accurate predictions to test data may be used for further model evaluation.
- A confusion matrix is created after model evaluation to calculate accuracy and precision numerically.
- After model evaluation, our model enters the final stage that is prediction.

6. Prediction:

- In the prediction phase developer deploys the model.
- After model deployment, it becomes ready to make predictions.
- Predictions are made on training data and test data to have a better understanding of the build model.

2. What is Cross Validation (CV) and resampling?

Resampling

- In machine learning, resampling is a method that involves repeatedly drawing samples from the training dataset. The samples are then used to refit a specific model to retrieve more information about the fitted model.

Some common techniques for resampling data include:

- **Cross-validation:** Splits the data into multiple subsets, and the model is trained and tested on each subset. This helps to estimate the model's performance on unseen data and reduces overfitting.
- **Boostrap:** Provides a measure of accuracy for a given parameter/method.

Cross Validation (CV)

- Cross-validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. We can also say that it is a technique to check how a statistical model generalizes to an independent dataset.

The basic steps of cross-validations are:

- Reserve a subset of the dataset as a validation set.
- Provide the training to the model using the training dataset.
- Now, evaluate model performance using the validation set. If the model performs well with the validation set, perform the further step, else check for the issues.

Methods used for Cross-Validation

There are some common methods that are used for cross-validation. These methods are given below:

6. Validation Set Approach
7. Leave-P-out cross-validation
8. Leave one out cross-validation
9. K-fold cross-validation
10. Stratified k-fold cross-validation

Validation Set Approach

- We divide our input dataset into a training set and test or validation set in the validation set approach. Both the subsets are given 50% of the dataset.
- But it has one of the big disadvantages that we are just using a 50% dataset to train our model, so the model may miss out to capture important information of the dataset. It also tends to give the underfitted model.

Leave-P-out cross-validation

- In this approach, the p datasets are left out of the training data. It means, if there are total n datapoints in the original input dataset, then n-p data points will be used as the training dataset and the p data points as the validation set. This complete process is repeated for all the samples, and the average error is calculated to know the effectiveness of the model.
- There is a disadvantage of this technique; that is, it can be computationally difficult for the large p.

Leave one out cross-validation

- This method is similar to the leave-p-out cross-validation, but instead of p, we need to take 1 dataset out of training. It means, in this approach, for each learning set, only one datapoint is reserved, and the remaining dataset is used to train the model.

This process repeats for each datapoint. Hence for n samples, we get n different training set and n test set. It has the following features:

- In this approach, the bias is minimum as all the data points are used.
- The process is executed for n times; hence execution time is high.
- This approach leads to high variation in testing the effectiveness of the model as we iteratively check against one data point.

3. What is K-fold CV and explain in detail.

With neat diagram explain about K-fold Cross Validation Technique. [Apr 2024]

K-Fold Cross-Validation

- K-fold cross-validation approach divides the input dataset into K groups of samples of equal sizes. These samples are called **folds**.
- For each learning set, the prediction function uses k-1 folds, and the rest of the folds are used for the test set. This approach is a very popular CV approach because it is easy to understand, and the output is less biased than other methods.

The steps for k-fold cross-validation are:

- Split the input dataset into K groups
- For each group:
  - Take one group as the reserve or test data set.
  - Use remaining groups as the training dataset
  - Fit the model on the training set and evaluate the performance of the model using the test set.
- Let's take an example of 5-folds cross-validation. So, the dataset is grouped into 5 folds. On 1st iteration, the first fold is reserved for test the model, and rest are used to train the model. On 2nd iteration, the second fold is used to test the model, and rest are used to train the model. This process will continue until each fold is not used for the test fold.

Consider the below diagram:

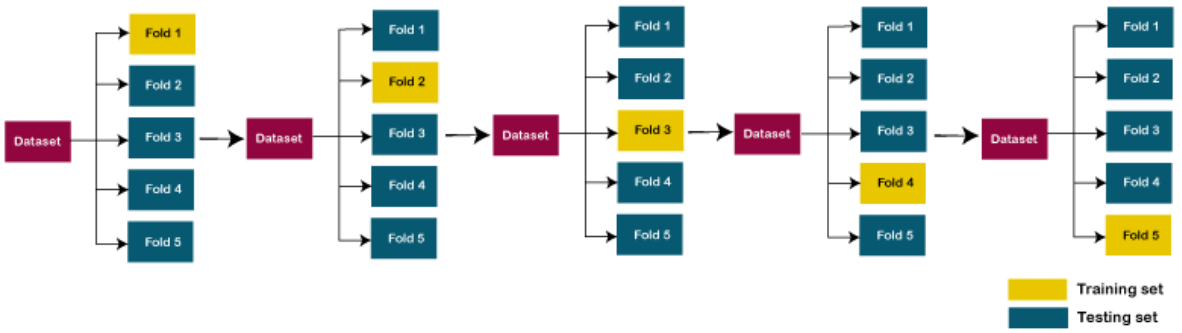


Figure 5.2 steps for k-fold cross-validation

4. Explain in detail about bootstrapping.

Bootstrapping

- Bootstrapping is a resampling technique in machine learning that involves repeatedly drawing samples from source data with replacement. The term comes from the idea of lifting oneself up without external help by pulling on one's own bootstraps.
- In the bootstrap, we sample N instances from a dataset of size N with replacement. The original dataset is used as the validation set. The probability that we pick an instance is 1/N; the probability that we do not pick it is 1 − 1/N. The probability that we do not pick it after N draws is

$$\left(1-\frac{1}{N}\right)^{N} \approx e^{-1} = 0.368$$

- This means that the training data contains approximately 63.2 percent of the instances; that is, the system will not have been trained on 36.8 percent of the data, and the error estimate will be pessimistic. The solution is replication, that is, to repeat the process many times and look at the average behavior.

Types of bootstrapping methods

- There are 2 types of bootstrapping methods that are applicable in statistics and Machine Learning.

Parametric Bootstrap Method

- In this method, the distribution parameter must be known. This means that the assumption of the kind of distribution the sample has must be provided beforehand.

- For instance, it must be known to the user if the sample has Gaussian Distribution or Skewed distribution. This type of bootstrap method is more efficient since it already knows the nature of distribution.

3. <u>Non-Parametric Bootstrap Method</u>

- Unlike the parametric bootstrap method, this type does not require the parameter of distribution to be known beforehand. Therefore, this type of bootstrap method works without assuming the nature of the sample distribution.

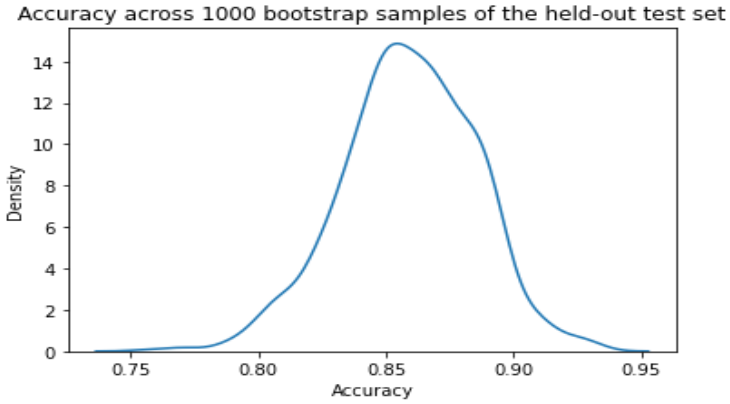


Figure 5.3 accuracy across 1000 bootstrap samples of the held-out test set

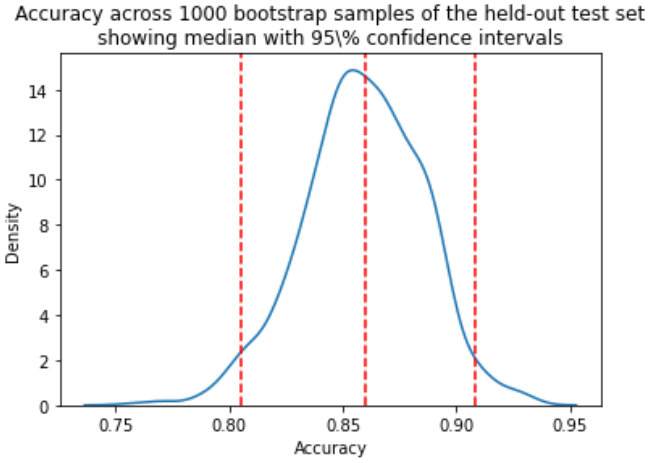


Figure 5.3 accuracy across 1000 bootstrap samples of the held-out test set showing median with 95% confidence intervals

5. Explain in detail about Measuring classifier performance and explain the performance metrics for Classification.

Measuring classifier performance

- Evaluating the performance of a Machine learning model is one of the important steps while building an effective ML model.
- ***To evaluate the performance or quality of the model, different metrics are used, and these metrics are known as performance metrics or evaluation metrics.***
- These performance metrics help us understand how well our model has performed for the given data.
- In this way, we can improve the model's performance by tuning the hyper-parameters.
- Each ML model aims to generalize well on unseen/new data, and performance metrics help determine how well the model generalizes on the new dataset.
- In machine learning, each task or problem is divided into **classification** and **Regression**. Not all metrics can be used for all types of problems; hence, it is important to know and understand which metrics should be used. Different evaluation metrics are used for both Regression and Classification tasks. In this topic, we will discuss metrics used for classification and regression tasks.

1. Performance Metrics for Classification

In a classification problem, the category or classes of data is identified based on training data. The model learns from the given dataset and then classifies the new data into classes or groups based on the training. It predicts class labels as the output, such as *Yes or No, 0 or 1, Spam or Not Spam,* etc. To evaluate the performance of a classification model, different metrics are used, and some of them are as follows:

- **Accuracy**
- **Confusion Matrix**
- **Precision**
- **Recall**
- **F-Score**
- **AUC(Area Under the Curve)-ROC**

I. Accuracy

The accuracy metric is one of the simplest Classification metrics to implement, and it can be determined as the number of correct predictions to the total number of predictions.

- It can be formulated as:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ number\ of\ predictions}$$

To implement an accuracy metric, we can compare ground truth and predicted values in a loop, or we can also use the scikit-learn module for this.

Firstly, we need to import the *accuracy_score* function of the scikit-learn library as follows:

from sklearn.metrics import accuracy_score

Here, metrics is a class of sklearn.

Then we need to pass the ground truth and predicted values in the function to calcul ate the accuracy.

print(f'Accuracy Score is {accuracy_score(y_test,y_hat)}')

II. Confusion Matrix

- A confusion matrix is a tabular representation of prediction outcomes of any binary classifier, which is used to describe the performance of the classification model on a set of test data when true values are known.
- The confusion matrix is simple to implement, but the terminologies used in this matrix might be confusing for beginners.
- A typical confusion matrix for a binary classifier looks like the below image(However, it can be extended to use for classifiers with more than two classes).

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

We can determine the following from the above matrix:

- In the matrix, columns are for the prediction values, and rows specify the Actual values. Here Actual and prediction give two possible classes, Yes or No. So, if we are predicting the presence of a disease in a patient, the Prediction column with Yes means, Patient has the disease, and for NO, the Patient doesn't have the disease.
- In this example, the total number of predictions are 165, out of which 110 time predicted yes, whereas 55 times predicted No.
- However, in reality, 60 cases in which patients don't have the disease, whereas 105 cases in which patients have the disease.

In general, the table is divided into four terminologies, which are as follows:

1. **True Positive(TP):** In this case, the prediction outcome is true, and it is true in reality, also.
2. True Negative(TN): in this case, the prediction outcome is false, and it is false in reality, also.
3. False Positive(FP): In this case, prediction outcomes are true, but they are false in actuality.
4. False Negative(FN): In this case, predictions are false, and they are true in actuality.

III. Precision

- The precision metric is used to overcome the limitation of Accuracy. The precision determines the proportion of positive prediction that was actually correct. It can be calculated as the True Positive or predictions that are actually true to the total positive predictions (True Positive and False Positive).

$$Precision = \frac{TP}{(TP + FP)}$$

IV. Recall or Sensitivity

- It is also similar to the Precision metric; however, it aims to calculate the proportion of actual positive that was identified incorrectly. It can be calculated as True Positive or predictions that are actually true to the total number of positives, either correctly predicted as positive or incorrectly predicted as negative (true Positive and false negative).
- The formula for calculating Recall is given below:

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

- *If we maximize precision, it will minimize the FP errors, and if we maximize recall, it will minimize the FN error.*

V. F-Scores

- F-score or F1 Score is a metric to evaluate a binary classification model on the basis of predictions that are made for the positive class. It is calculated with the help of Precision and Recall. It is a type of single score that represents both Precision and Recall. So, ***the F1 Score can be calculated as the harmonic mean of both precision and Recall, assigning equal weight to each of them.***
- The formula for calculating the F1 score is given below:

$$F1 - score = 2 * \frac{precision * recall}{precision + recall}$$

VI. AUC-ROC

- Sometimes we need to visualize the performance of the classification model on charts; then, we can use the AUC-ROC curve. It is one of the popular and important metrics for evaluating the performance of the classification model.
- Firstly, let's understand ROC (Receiver Operating Characteristic curve) curve. ***ROC represents a graph to show the performance of a classification model at different threshold levels***. The curve is plotted between two parameters, which are:
    - **True Positive Rate**
    - **False Positive Rate**
- TPR or true Positive rate is a synonym for Recall, hence can be calculated as:

$$TPR = \frac{TP}{TP + FN}$$

FPR or False Positive Rate can be calculated as:

$$TPR = \frac{FP}{FP + TN}$$

AUC: Area Under the ROC curve

- AUC is known for **Area Under the ROC curve**. As its name suggests, AUC calculates the two-dimensional area under the entire ROC curve, as shown below image:

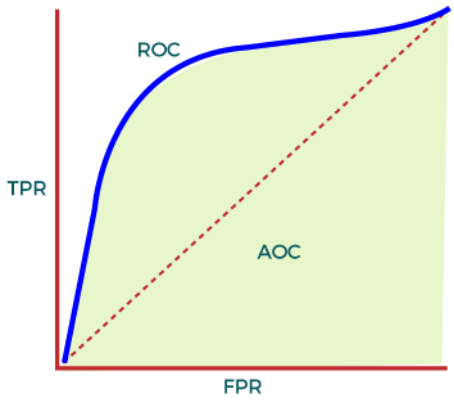


Fig.5.4 Area Under the ROC Curve

- AUC calculates the performance across all the thresholds and provides an aggregate measure. The value of AUC ranges from 0 to 1. It means a model with 100% wrong prediction will have an AUC of 0.0, whereas models with 100% correct predictions will have an AUC of 1.0.

6. Explain in detail about assessing a single classification algorithm and comparing two classification algorithms.

Hypothesis Testing

- Hypothesis testing is a common method to evaluate the performance and validity of machine learning models. It involves making a claim about a population parameter, such as the mean or the proportion, based on a sample of data.
- We define a statistic that obeys a certain distribution if the hypothesis is correct. If the statistic calculated from the sample has very low probability of being drawn from this distribution, then we reject the hypothesis; otherwise, we fail to reject it.

Level of significance:

- Refers to the degree of significance in which we accept or reject the null-hypothesis. 100% accuracy is not possible for accepting or rejecting a hypothesis, so we therefore select a level of significance that is usually 5%.

- This is normally denoted with alpha(maths symbol) and generally it is 0.05 or 5% , which means your output should be 95% confident to give similar kind of result in each sample.
- **Type I error:** When we reject the null hypothesis, although that hypothesis was true. Type I error is denoted by alpha. In hypothesis testing, the normal curve that shows the critical region is called the alpha region
- **Type II errors:** When we accept the null hypothesis but it is false. Type II errors are denoted by beta. In Hypothesis testing, the normal curve that shows the acceptance region is called the beta region.
- **One tailed test :-** A test of a statistical hypothesis , where the region of rejection is on only **one** side of the sampling distribution , is called a **one-tailed test**.

	Decision	
Truth	Fail to reject	Reject
True	Correct	Type I error
False	Type II error	Correct (power)

Figure 5. Table of Type I error, Type II error and power of a test

There are two types of hypotheses: the null hypothesis (H0) and the alternative hypothesis (H1 or Ha).

- **Null Hypothesis(H0):** This hypothesis suggests that there is no significant difference or effect, and any observed results are due to chance. It often represents the status quo or a baseline assumption.
- **Alternative Hypothesis(H1 or Ha):** This hypothesis contradicts the null hypothesis, proposing that there is a significant difference or effect in the population. It is what researchers aim to support with evidence.

Assessing a single classification algorithm

Binomial test

- *"The binomial test checks whether the frequency distribution of a variable with two values/categories in the sample corresponds to the distribution in the population."*
- The binomial test is a hypothesis test used when there is a categorical variable with two expressions, e.g., *gender* with "male" and "female". The binomial test can then check whether the frequency distribution of the variable corresponds to an expected distribution.

- We have a ***single training set T*** and ***a single validation set V.***
- We train our classifier on T and test it on V.
- We denote by ***p the probability*** that the classifier makes a misclassification error.
- We do not know p; it is what we would like to estimate or test a hypothesis about.
- On the instance with ***index t*** from the validation set V, let us say ***xt*** denotes the ***correctness of the classifier's decision***: ***xt is a 0/1*** Bernoulli random variable that takes the value 1 when the classifier commits an error and 0 when the classifier is correct.
- The binomial random variable X denotes the total number of errors:

$$X = \sum_{t=1}^{N} x^t$$

Assessing Error:

- We would like to test whether the error probability p is less than or equal to some value p0 we specify:

  $H_0: p \leq p_0$ **vs.** $H_1: p > p_0$

  If the probability of error is p, the probability that the classifier commits j errors out of N is

$$P\{X = j\} = \binom{N}{j} p^j (1-p)^{N-j}$$

- It is reasonable to reject p ≤ p0 if in such a case, the probability that binomial test we see X = e errors or more is very unlikely. That is, the binomial test rejects the hypothesis if

$$P\{X \geq e\} = \sum_{x=e}^{N} \binom{N}{x} p_0^x (1-p_0)^{N-x} < \alpha$$

where α is the significance, for example, 0.05.

Approximate Normal Test

- *Number of errors X is approx N with mean Np_0 and var $Np_0(1-p_0)$*

$$\frac{X - Np_0}{\sqrt{Np_0(1-p_0)}} \sim Z$$

Accept if this prob for X = e is less than z_{1-a}

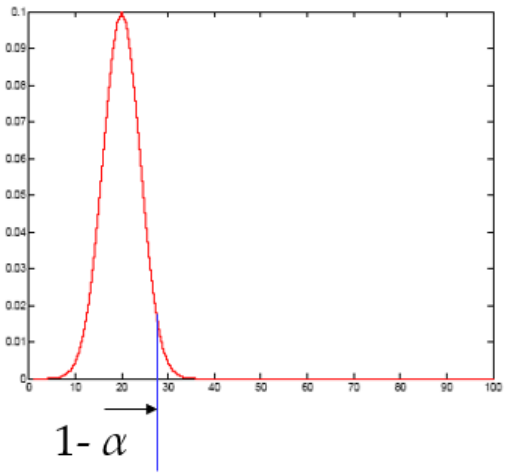


Fig.5.5 Approximate Normal Test

7. Explain in detail about t test, McNemar's test, K-fold CV paired t test.

Elaborate the t test, McNemars test and K-Fold CV paired t test by giving your own example. [Apr 2024]

T Test

1. t-test (Student's t-test)

- A **t-test** is a statistical test used to determine whether there is a significant difference between the means of two groups. It assumes the data is normally distributed and that the variance between the groups is approximately equal (depending on the test variant).

Types of t-tests:

1. **Independent (Unpaired) t-test**
    - Compares the means of two independent groups.
    - Used when we have two different samples (e.g., control vs. treatment group).
    - Assumptions:
        - Data is normally distributed.
        - Groups have equal variances (if not, Welch's t-test is used).
        - Observations are independent.
2. **Paired t-test (Dependent t-test)**

- Compares the means of the same group at different times or under different conditions.
- Used in **pre-test/post-test** scenarios.
- Assumptions:
  - The differences between paired observations follow a normal distribution.
  - The samples are dependent (i.e., the same subjects are used before and after).

3. **One-sample t-test**
   - Compares the mean of a single sample to a known population mean.
   - Assumptions:
     - The sample is randomly drawn from a normally distributed population.

Formula for t-test:

For an independent t-test:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

where:

- $\bar{X}_1, \bar{X}_2$ = means of the two samples
- s_1^2, s_2^2 = variances of the two samples
- n_1, n_2 = sample sizes of the two groups

For a paired t-test:

$$t = \frac{\bar{D}}{s_D/\sqrt{n}}$$

where:

- $\bar{D}$ = mean of the differences
- s_D = standard deviation of the differences
- n = number of pairs

When to Use a t-test?

- Comparing means of two small groups (n < 30).
- Data is approximately normally distributed.
- Comparing two groups only (not more).

2. McNemar's Test

- McNemar's test is a non-parametric test used to analyze paired nominal (categorical) data. It determines whether there is a significant change in proportions before and after a treatment or intervention.

When to Use McNemar's Test?

- When data is paired and binary (e.g., Yes/No, Success/Failure).
- Typically used in medical studies, classification problems (before and after applying a model), or A/B testing when analyzing matched subjects.

Example Use Case:

- Imagine testing a new diagnostic test. Suppose 100 patients were tested with both the old and new tests. The results are recorded in a 2x2 contingency table:

Example Use Case:

Imagine testing a **new diagnostic test**. Suppose 100 patients were tested with both the old and new tests. The results are recorded in a **2x2 contingency table**:

	New Test Positive	New Test Negative
Old Test Positive	a	b
Old Test Negative	c	d

- **McNemar's test** focuses on **b and c** (discordant pairs), where **one** test changed but not both.

Formula:

$$\chi^2 = \frac{(b-c)^2}{b+c}$$

- If b and c are very different, it indicates a significant change.

McNemar's Test Assumptions:

- Data is paired (dependent).
- The variable of interest is **dichotomous** (binary).
- Sample size should be large enough ($b + c \geq 10$) for the chi-square approximation to hold.

Variants:

- Exact McNemar's Test (if sample is small, uses binomial distribution instead of chi-square).
- McNemar-Bowker Test (used for multi-category data).

3. K-Fold Cross-Validation Paired t-test

- A **K-fold cross-validation paired t-test** is a statistical method used to compare the performance of two machine learning models over multiple iterations of **K-fold cross-validation**.

Steps:

1. Split data into **K folds** (e.g., 5-fold CV).
2. Train and test both models on each fold.
3. Collect performance metrics (e.g., accuracy, RMSE) for both models across all folds.

4. Conduct a **paired t-test** to compare model performance.

Why Use a Paired t-test?

- Since the same data is used in both models (paired), an **independent t-test** is inappropriate.
- A **paired t-test** ensures the comparison accounts for the dependency between folds.

Formula:

$$t = \frac{\bar{D}}{s_D/\sqrt{K}}$$

where:

- $\bar{D}$ = mean difference in performance across K folds.
- s_D = standard deviation of differences.
- K = number of folds.

Example:

Suppose we evaluate **Model A and Model B** on 5-fold CV:

Fold	Model A Accuracy	Model B Accuracy	Difference (D)
1	85%	83%	2%
2	87%	84%	3%
3	86%	82%	4%
4	84%	83%	1%
5	88%	85%	3%

If the **paired t-test p-value** is < 0.05, we conclude that one model **significantly outperforms** the other.

Advantages:

- Reduces variance from data partitioning (compared to a simple train-test split).
- Accounts for dependency between models.

Assumptions:

- The differences between model performances **follow a normal distribution**.
- K-fold CV results are **independent across folds** (this may be violated in time series or highly correlated data).
- **Summary**

Test	Purpose	Assumptions	Example Use
t-test	Compares means of two groups	Normal distribution, equal variance (for independent t-test)	A/B testing on website performance
McNemar's Test	Compares paired categorical data	Binary outcomes, paired samples	Comparing before-after performance of a diagnostic test
K-fold CV Paired t-test	Compares two models across K-fold cross-validation	Performance differences are normally distributed	Comparing two machine learning models across multiple folds

Each test is useful depending on data type (continuous vs categorical) and study design (independent vs paired).

8. Explain in detail about comparing two classification algorithms. [Apr 2024]

- When comparing two classification algorithms, the goal is to determine which model performs better on a given dataset. The comparison can be based on various performance metrics, statistical tests, and validation strategies.

1. Key Considerations When Comparing Classifiers

Before comparing two classifiers, we must consider:

- **Evaluation Metrics** (accuracy, precision, recall, F1-score, AUC-ROC, etc.).
- **Validation Strategy** (train-test split, k-fold cross-validation, repeated k-fold CV).
- **Statistical Significance Testing** (paired t-test, McNemar's test, Wilcoxon signed-rank test).
- **Dataset Properties** (imbalanced data, feature selection, etc.).

2. Evaluation Metrics for Classification

Different metrics are used to evaluate classification models based on the nature of the problem.

Confusion Matrix

A **confusion matrix** summarizes the classification results in a **binary** or **multi-class** setting.

Actual \ Predicted	**Positive (1)**	**Negative (0)**
Positive (1)	True Positive (TP)	False Negative (FN)
Negative (0)	False Positive (FP)	True Negative (TN)

Common Performance Metrics:

1. **Accuracy**

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

- Works well for balanced datasets but can be misleading for imbalanced datasets.

2. **Precision (Positive Predictive Value - PPV)**

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

- Important for minimizing false positives (e.g., fraud detection).

3. **Recall (Sensitivity, True Positive Rate - TPR)**

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

- Important for minimizing false negatives (e.g., disease diagnosis).

4. **F1-Score (Harmonic Mean of Precision & Recall)**

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

- A balanced measure when both precision and recall are important.

5. **ROC Curve & AUC (Area Under Curve)**

- Measures the trade-off between **TPR** (Recall) and **FPR** (False Positive Rate).
- AUC (Area Under Curve) closer to **1.0** indicates better model performance.

6. **Log Loss (Cross-Entropy Loss)**

$$LogLoss = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$$

- Used when models output **probabilities** instead of class labels.

3. Validation Strategies

To ensure a fair comparison, models should be evaluated on the same dataset using proper validation techniques.

(A) Train-Test Split

- Splitting the dataset into a **training set** and **test set** (e.g., 80%-20%).
- Simple but may lead to biased estimates if the dataset is small.

(B) k-Fold Cross-Validation (CV)

- The dataset is split into **k** subsets (folds).
- Each model is trained on **k-1 folds** and tested on the remaining fold.
- Common values: **k = 5 or 10**.
- Reduces the risk of overfitting.

(C) Stratified k-Fold Cross-Validation

- Ensures class distribution is maintained in each fold (useful for imbalanced datasets).

(D) Leave-One-Out Cross-Validation (LOOCV)

- Each instance is used once as a test set while all others form the training set.
- Computationally expensive for large datasets.

(E) Repeated k-Fold Cross-Validation

- Repeats **k-fold CV multiple times** with different random splits for better robustness.

4. Statistical Significance Tests for Comparison

Since a classification model's performance varies due to random initialization, dataset splits, and sample variations, statistical tests help determine if one model is **truly better**.

(A) Paired t-Test (for k-Fold Cross-Validation Results)

- Compares the mean performance of two classifiers across **multiple folds**.
- Assumes that performance differences follow a normal distribution.
- Formula: t=D¯sD/kt = \frac{\bar{D}}{s_D / \sqrt{k}}

$$t = \frac{\bar{D}}{s_D/\sqrt{k}}$$

where:

- $\bar{D}$ = mean difference in performance across **k** folds.
- s_D = standard deviation of differences.
- k = number of folds.

(B) McNemar's Test (for Paired Binary Classifications)

- Used when the same instances are **classified differently** by two models.
- Based on a **contingency table** of classification disagreements:

	Model B Correct	**Model B Incorrect**
Model A Correct	a	b
Model A Incorrect	c	d

- McNemar's test statistic:

$$\chi^2 = \frac{(b-c)^2}{b+c}$$

  - If **b** ≠ **c**, one model is significantly better.

When to Use?

- For comparing two classifiers trained on the same dataset.
- Works best when the focus is on classification disagreements.

(C) Wilcoxon Signed-Rank Test (Non-Parametric)

- A non-parametric test for comparing paired results.
- Useful when data does not follow a normal distribution.
- Based on ranking performance differences across k-folds and evaluating their sum.

When to Use?

- If the paired t-test assumptions are violated.
- If performance differences are not normally distributed.

5. Example: Comparing Two Classification Algorithms in Python

- We compare **Logistic Regression** and **Random Forest** using **10-fold CV** and a paired t-test.
- We compare **Logistic Regression** and **Random Forest** using **10-fold CV** and a paired t-test.

Implementation

```
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from scipy.stats import ttest_rel
from sklearn.datasets import make_classification
from sklearn.model_selection import StratifiedKFold
# Generate synthetic dataset
X, y = make_classification(n_samples=1000, n_features=20, random_state=42)
# Define models
model1 = LogisticRegression()
model2 = RandomForestClassifier()
# 10-fold Stratified Cross-Validation
kf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
# Evaluate models
scores1 = cross_val_score(model1, X, y, cv=kf, scoring='accuracy')
scores2 = cross_val_score(model2, X, y, cv=kf, scoring='accuracy')
# Paired t-test
t_stat, p_value = ttest_rel(scores1, scores2)
```

```
print(f"Model 1 Mean Accuracy: {scores1.mean():.4f}")
print(f"Model 2 Mean Accuracy: {scores2.mean():.4f}")
print(f"T-statistic: {t_stat:.4f}, P-value: {p_value:.4f}")
# Interpretation
alpha = 0.05
if p_value < alpha:
    print("Significant difference between models.")
else:
    print("No significant difference between models.")
```

Summary

Comparison Method	**When to Use?**
Accuracy, F1-score, AUC	Basic performance comparison
k-Fold CV	More reliable estimate of model performance
Paired t-Test	When using k-fold CV results
McNemar's Test	When analyzing classification disagreements
Wilcoxon Signed-Rank Test	When performance differences are not normally distributed

9. Consider the following list that contains name,age,gender, and class of sports. In the gender field males are denoted by the numeric value 0 and formulae by 1. using the K nearest neighbor algorithm find class of sports for a girl whose name is Angelina, her k factor is 3, and her age is 5. [Apr 2024]

Ajay	32	0	Football
Mark	40	0	Neither
Sara	16	1	Crivket
Zaira	34	1	Cricket
Sachin	55	0	Neither
Ragul	40	0	Cricket
Pooja	20	1	Neither
Smith	15	0	Cricket

Laxmi	55	1	Football
Michael	15	0	Football

Step 1: Convert Data into Numerical Format

Name	Age	Gender	Class of Sports
Ajay	32	0	Football
Mark	40	0	Neither
Sara	16	1	Cricket
Zaira	34	1	Cricket
Sachin	55	0	Neither
Ragul	40	0	Cricket
Pooja	20	1	Neither
Smith	15	0	Cricket
Laxmi	55	1	Football
Michael	15	0	Football

Step 2: Calculate Euclidean Distance

Euclidean Distance formula:

d=(x2−x1)2+(y2−y1)2d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}d=(x2−x1)2+(y2−y1)2

where:

- x1,y1x_1, y_1x1,y1 are Angelina's data: (5,1)(5, 1)(5,1)
- x2,y2x_2, y_2x2,y2 are other entries.

Name	Age	Gender	Distance Calculation	Distance
Ajay	32	0	$\sqrt{(32-5)^2+(0-1)^2}$	27.02
Mark	40	0	$\sqrt{(40-5)^2+(0-1)^2}$	35.01
Sara	16	1	$\sqrt{(16-5)^2+(1-1)^2}$	11.00
Zaira	34	1	$\sqrt{(34-5)^2+(1-1)^2}$	29.00
Sachin	55	0	$\sqrt{(55-5)^2+(0-1)^2}$	50.01
Ragul	40	0	$\sqrt{(40-5)^2+(0-1)^2}$	35.01
Pooja	20	1	$\sqrt{(20-5)^2+(1-1)^2}$	15.00
Smith	15	0	$\sqrt{(15-5)^2+(0-1)^2}$	10.05
Laxmi	55	1	$\sqrt{(55-5)^2+(1-1)^2}$	50.00
Michael	15	0	$\sqrt{(15-5)^2+(0-1)^2}$	10.05

Step 3: Select the 3 Nearest Neighbors (k=3)

Sorting distances:

1. Smith (10.05) → Cricket
2. Michael (10.05) → Football
3. Sara (11.00) → Cricket

Step 4: Classify Based on Majority Vote

- **Cricket:** 2 votes (Smith, Sara)
- **Football:** 1 vote (Michael)

Since **Cricket** has the majority, **Angelina will be classified into "Cricket"**.

Final Answer:

Angelina's predicted class of sports is "Cricket"

10. The grades of a class of 9 students on a midterm report (X) and on the final examination (Y) are as follows

X	**77**	**50**	**71**	**72**	**81**	**94**	**96**	**99**	**67**
Y	**82**	**66**	**78**	**34**	**47**	**85**	**99**	**99**	**68**

Estimate the linear regression line.

Estimate the final estimation grade of a student who received a grade of 85 on the midterm report. [Apr 2024]

To estimate the linear regression line for the given data and predict the final examination grade for a student who scored 85 on the midterm, follow these steps:

Given Data:

Student	Midterm Grade (X)	Final Grade (Y)
1	77	82
2	50	66
3	71	78
4	72	34
5	81	47
6	94	85
7	96	99
8	99	99
9	67	68

Step 1: Calculate Means

- Mean of X ($\bar{X}$):

$$\bar{X} = \frac{77 + 50 + 71 + 72 + 81 + 94 + 96 + 99 + 67}{9} = \frac{707}{9} \approx 78.56$$

- Mean of Y ($\bar{Y}$):

$$\bar{Y} = \frac{82 + 66 + 78 + 34 + 47 + 85 + 99 + 99 + 68}{9} = \frac{658}{9} \approx 73.11$$

Step 2: Calculate Slope (b) and Intercept (a)

- **Slope (b):**

$$b = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sum(X_i - \bar{X})^2}$$

First, compute the necessary sums:

$$\sum(X_i - \bar{X})(Y_i - \bar{Y}) = (77 - 78.56)(82 - 73.11) + (50 - 78.56)(66 - 73.11) + \ldots + (67 - 78.56)(68 - 73$$

$$\sum(X_i - \bar{X})^2 = (77 - 78.56)^2 + (50 - 78.56)^2 + \ldots + (67 - 78.56)^2$$

After calculating these sums:

$$b \approx \frac{-1.56 \times 8.89 + (-28.56) \times (-7.11) + \ldots + (-11.56) \times (-5.11)}{(-1.56)^2 + (-28.56)^2 + \ldots + (-11.56)^2}$$

Simplifying further:

$$b \approx \frac{59184 - \frac{707 \times 658}{9}}{57383 - \frac{707^2}{9}}$$

$$b \approx \frac{59184 - 51614.22}{57383 - 55680.11} \approx \frac{7569.78}{1702.89} \approx 4.45$$

- **Intercept (a):**

$$a = \bar{Y} - b \times \bar{X}$$

$$a \approx 73.11 - 4.45 \times 78.56 \approx 73.11 - 349.58 \approx -276.47$$

Regression Equation:

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

$$\hat{Y} = -276.47 + 4.45X$$

Step 3: Predict Final Grade for Midterm Score of 85

$$\hat{Y} = -276.47 + 4.45 \times 85$$

$$\hat{Y} = -276.47 + 378.25 \approx 101.78$$

Conclusion:

- Based on the linear regression model, a student who scored 85 on the midterm is predicted to score approximately **101.78** on the final examination. However, since exam scores are typically capped at 100, this suggests the student is expected to achieve a top score on the final exam.

11.

Cases	Actual Sick?	Predicted Sick?
1	1	1
2	1	0
3	0	1
4	0	0
5	1	1
6	0	0
7	1	0
8	0	1
9	1	1
10	0	0

Construct confusion matrix for the above table. Calculate accuracy, precision, recall and F1 score. The values in the table denotes 1 for Yes and 0 for No

Explain why accuracy is generally not the preferred performance measure for classifiers. Especially when you are dealing with skewed datasets(i.e. when some classes are much more frequent than others) [Nov 2024]

Let's construct the confusion matrix based on the given data:

Actual / Predicted	**Predicted Sick (1)**	**Predicted Not Sick (0)**
Actual Sick (1)	True Positives (TP) = 3	False Negatives (FN) = 2
Actual Not Sick (0)	False Positives (FP) = 2	True Negatives (TN) = 3

Now, let's calculate the performance metrics:

1. **Accuracy**

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

Measures how often the model makes correct predictions.

2. **Precision (Positive Predictive Value)**

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

Measures how many of the predicted positive cases are actually positive.

3. **Recall (Sensitivity or True Positive Rate)**

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

Measures how many actual positive cases were correctly identified.

4. **F1 Score** (Harmonic Mean of Precision and Recall)

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

A balanced measure considering both Precision and Recall.

Why Accuracy is Not the Preferred Metric for Skewed Datasets

Accuracy is not always the best measure for evaluating classifier performance, especially when dealing with **imbalanced datasets**. Here's why:

1. **Misleading Performance in Imbalanced Data**
   - Suppose 90% of the samples belong to class 0 (not sick) and only 10% belong to class 1 (sick).

- A classifier that always predicts "Not Sick" (0) would still achieve 90% accuracy despite failing to detect any of the actual sick cases.

2. **Does Not Consider False Positives and False Negatives Separately**
   - In medical diagnosis, misclassifying a sick patient as healthy (False Negative) is much worse than misclassifying a healthy person as sick (False Positive).
   - Accuracy does not distinguish between these two types of errors.
3. **Precision, Recall, and F1 Score are More Informative**
   - **Precision** helps when False Positives are costly (e.g., unnecessary medical treatments).
   - **Recall** helps when False Negatives are costly (e.g., missing a serious disease).
   - **F1 Score** balances both, making it a better metric in cases of class imbalance.

Thus, in skewed datasets, accuracy can give a **false sense of model performance**, while metrics like Precision, Recall, and F1 Score provide a more **realistic** evaluation.

11. (i)Among 100 person tested for cancer, actually (truly) 70 are affected by cancer. In the model prediction, 60 persons are predicted as cancer in which 15 are not affected by cancer (It means 15 persons are wrongly predicted as cancer). Construct confusion matrix and calculate precision and recall.

(ii) Illustrate the need for precision and recall.

Step 1: Constructing the Confusion Matrix

We are given the following information:

- **Total tested persons = 100**
- **Actual cancer cases = 70 (Truly Sick)**
- **Predicted cancer cases = 60**
- **False Positives (FP) = 15** (People who were predicted as cancer but are actually not affected)
- **True Positives (TP) = 60 - 15 = 45** (Correctly predicted cancer cases)
- **False Negatives (FN) = 70 - 45 = 25** (Actual cancer patients who were wrongly predicted as not having cancer)
- **True Negatives (TN) = 100 - (TP + FP + FN) = 100 - (45 + 15 + 25) = 15** (Correctly predicted non-cancer cases)

Now, the confusion matrix:

Actual / Predicted	Predicted Cancer (1)	Predicted Not Cancer (0)
Actual Cancer (1)	TP = 45	FN = 25
Actual Not Cancer (0)	FP = 15	TN = 15

Step 2: Calculating Precision and Recall

- **Precision (Positive Predictive Value)**

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{45}{45 + 15} = \frac{45}{60} = 0.75 \text{ (or 75%)}$$

Interpretation: Among all the people the model predicted as having cancer, **75% were actually affected**.

- **Recall (Sensitivity or True Positive Rate)**

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{45}{45 + 25} = \frac{45}{70} \approx 0.6429 \text{ (or 64.3%)}$$

Interpretation: The model correctly identified **64.3% of actual cancer cases**.

Step 3: Why Precision and Recall Are Important?

- **Need for Precision**
  - Precision is important when **False Positives (FP) are costly**.
  - Example: If a cancer screening test has low precision, many healthy people may be wrongly diagnosed with cancer, leading to unnecessary stress, further tests, and treatments.
- **Need for Recall**
  - Recall is important when **False Negatives (FN) are costly**.
  - Example: If a cancer screening test has low recall, many actual cancer patients may be missed, delaying their treatment and reducing their chances of recovery.

Trade-off Between Precision and Recall

- A **high precision** model ensures fewer false positives but might miss some actual positive cases (low recall).
- A **high recall** model catches most actual positives but may have more false positives (low precision).
- The choice between precision and recall depends on the **application and consequences** of errors.
  - **Medical diagnosis → Prioritize Recall (to catch as many true cases as possible).**
  - **Spam detection → Prioritize Precision (to avoid flagging legitimate emails as spam).**

12. Consider a function f(x) =x2+2x+3, a data scientist wishes to find the minimum of this fiunction using gradient descent. He uses the following update rule.

X(t+1)=x(t)-(2x(t)+2)

Given this information, answer questions (a) to (d).

(a) If the expression for the gradient is correct, what is the step size that the data scientist is using?
(b) If x(0)=0, what is the value of x(100)?
(c) Justify why gradient descent update rule wont reach the minimum of the function f(x)?
(d) Using gradient descent with appropriate step size, find the value of x, at which the function attains its minimum value, if such a minimum exists.

[Nov 2024]

We are given the function:

$$f(x) = x^2 + 2x + 3$$

Step 1: Compute the Gradient

The derivative (gradient) of $f(x)$ is:

$$\frac{d}{dx} f(x) = 2x + 2$$

The gradient descent update rule provided is:

$$x_{t+1} = x_t - (2x_t + 2)$$

(a) Step Size Calculation

Gradient descent generally follows the update rule:

$$x_{t+1} = x_t - \alpha \cdot \nabla f(x_t)$$

Comparing with our update rule:

$$x_{t+1} = x_t - (2x_t + 2)$$

It implies that the step size α is **1**, because the update term exactly matches the gradient without any additional scaling factor.

Thus, **the step size used is** $\alpha = 1$.

(b) Compute x_{100} Given $x_0 = 0$

Using the given update rule:

$$x_{t+1} = x_t - (2x_t + 2)$$

Substituting $x_0 = 0$:

$$x_1 = 0 - (2(0) + 2) = 0 - 2 = -2$$

$$x_2 = -2 - (2(-2) + 2) = -2 - (-4 + 2) = -2 - (-2) = 0$$

Observing the pattern:

- $x_0 = 0$
- $x_1 = -2$
- $x_2 = 0$
- $x_3 = -2$
- $x_4 = 0$

- $x_5 = -2$

The sequence alternates between 0 and -2. Since x_{100} is even, it follows the same pattern as x_0, meaning:

$$x_{100} = 0$$

(c) Why Won't Gradient Descent Reach the Minimum?

The function $f(x) = x^2 + 2x + 3$ is a quadratic function with a **unique minimum** at:

$$\frac{d}{dx} f(x) = 2x + 2 = 0 \quad \Rightarrow \quad x = -1$$

However, with the given update rule, we see that the values of x_t keep oscillating between 0 and -2 rather than converging to $x = -1$. This happens because:

1. **Step size is too large**: The step size $\alpha = 1$ is too aggressive, causing the updates to overshoot the minimum.
2. **No convergence**: Instead of moving towards $x = -1$, the updates keep jumping back and forth between 0 and -2.

Thus, the given gradient descent **never reaches the true minimum**.

(d) Finding the Correct Step Size

For gradient descent to converge to the minimum, the step size α should be chosen appropriately.

The update rule should be:

$$x_{t+1} = x_t - \alpha(2x_t + 2)$$

To ensure convergence, the step size must satisfy:

$$0 < \alpha < \frac{2}{L}$$

where L is the **Lipschitz constant**, which for a quadratic function $f(x) = ax^2 + bx + c$ is simply the coefficient of x^2 multiplied by 2. Here, $L = 2$, so:

$$0 < \alpha < \frac{2}{2} = 1$$

Thus, an appropriate choice is $0 < \alpha < 1$. A common safe choice is $\alpha = 0.5$.

If we use $\alpha = 0.5$, the update rule becomes:

$$x_{t+1} = x_t - 0.5(2x_t + 2)$$

$$= x_t - (x_t + 1)$$

$$= -1$$

Since every update immediately reaches $x = -1$, **the function attains its minimum at** $x = -1$.

Final Answers

(a) **Step size** $\alpha = 1$.

(b) $x_{100} = 0$ (values alternate between 0 and -2).

(c) The update rule does not converge because the step size is too large, causing oscillations instead of moving toward the minimum.

(d) The minimum occurs at $x = -1$, and choosing a proper step size $0 < \alpha < 1$ ensures convergence.

13. Consider the following linear regression problem, use gradient descent to find the value of wi for two iteration.

Y=X*wi+b

Where y is the output of the model, X is input of the model.

Initial value of wi=4, b=5, alpha = 0.5

Consider the data points (5,14),(10,24),(15,32) for X and Y respectively

Use Mean Square Error as loss function. **[Nov 2024]**

Step 1: Define the Mean Squared Error (MSE) Loss Function

The Mean Squared Error (MSE) is given by:

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

where:

- $\hat{y}_i = X_i \cdot w_i + b$ (predicted value)
- y_i is the actual value
- $n = 3$ (since we have 3 data points)

The gradient descent update rule for w_i is:

$$w_i^{(t+1)} = w_i^{(t)} - \alpha \frac{\partial L}{\partial w_i}$$

where the gradient is:

$$\frac{\partial L}{\partial w_i} = -\frac{2}{n}\sum_{i=1}^{n} X_i(y_i - \hat{y}_i)$$

Step 2: Compute Initial Predictions

Given initial values:

- $w_i = 4, b = 5, \alpha = 0.5$
- Data points: $(X, Y) = \{(5, 14), (10, 24), (15, 32)\}$

Calculate $\hat{y}_i$ for each data point:

$$\hat{y}_1 = 5(4) + 5 = 20 + 5 = 25$$

$$\hat{y}_2 = 10(4) + 5 = 40 + 5 = 45$$

$$\hat{y}_3 = 15(4) + 5 = 60 + 5 = 65$$

Compute residuals $(y_i - \hat{y}_i)$:

Compute residuals $(y_i - \hat{y}_i)$:

$$e_1 = 14 - 25 = -11, \quad e_2 = 24 - 45 = -21, \quad e_3 = 32 - 65 = -33$$

Step 3: Compute Gradient

$$\frac{\partial L}{\partial w_i} = -\frac{2}{3}\sum_{i=1}^{3} X_i(y_i - \hat{y}_i)$$

$$= -\frac{2}{3}[(5)(-11) + (10)(-21) + (15)(-33)]$$

$$= -\frac{2}{3}[-55 - 210 - 495]$$

$$= -\frac{2}{3}(-760)$$

$$= \frac{1520}{3} = 506.67$$

Step 4: Update w_i for First Iteration

Using the gradient descent update rule:

$$w_i^{(1)} = 4 - (0.5 \times 506.67)$$

$$= 4 - 253.33$$

$$= -249.33$$

Step 5: Compute Second Iteration

Using $w_i^{(1)} = -249.33$, compute new predictions:

$$\hat{y}_1 = 5(-249.33) + 5 = -1246.67 + 5 = -1241.67$$

$$\hat{y}_2 = 10(-249.33) + 5 = -2493.33 + 5 = -2488.33$$

$$\hat{y}_3 = 15(-249.33) + 5 = -3740 + 5 = -3735$$

New residuals:

$$e_1 = 14 - (-1241.67) = 1255.67, \quad e_2 = 24 - (-2488.33) = 2512.33, \quad e_3 = 32 - (-3735) = 3767$$

New gradient:

$$\frac{\partial L}{\partial w_i} = -\frac{2}{3}[(5)(1255.67) + (10)(2512.33) + (15)(3767)]$$

$$= -\frac{2}{3}[6278.33 + 25123.33 + 56505]$$

$$= -\frac{2}{3}(87906.67)$$

$$= -58598$$

Update w_i:

$$w_i^{(2)} = -249.33 - (0.5 \times -58598)$$

$$= -249.33 + 29299$$

$$= 29049.67$$

Final Answer

- First iteration: $w_i^{(1)} = -249.33$
- Second iteration: $w_i^{(2)} = 29049.67$

This shows that **the learning rate (step size) is too high**, causing instability in gradient descent, as the weight oscillates wildly instead of converging. A smaller α (e.g., 0.01) would improve convergence.

Reg. No. : E N G G T R E E . C O M

Question Paper Code : 20070

B.E./B.Tech. DEGREE EXAMINATIONS, NOVEMBER/DECEMBER 2023.

Fourth Semester

Artificial Intelligence and Data Science

AL 3451 — MACHINE LEARNING

(Regulations 2021)

(Common to : Computer Science and Business Systems)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What is the objectives of machine learning.

2. Mention the various features of Machine Learning.

3. Relate entropy and information gain.

4. How does CART solve the regression problems?

5. Identify the challenges of the clustering algorithm.

6. Distinguish between Bagging and Boosting.

7. List out significant parts of biological neuron

8. Give the types of artificial neural networks.

9. List the various phases of the machine learning life cycle.

10. What is meant by cross-validation and resampling?

PART B — (5 × 13 = 65 marks)

11. (a) Examine in detail about Inductive bias with a suitable sketch. (13)

Or

(b) Elucidate the Vapnik-Chervonenkis (VC) dimension with suitable examples. (13)

12. (a) Write short notes on

(i) Regression and Correlation (6)

(ii) Limitation of Regression model. (7)

Or

(b) List the advantages of SVM and how optimal Hyperplane differs from Hyperplane. (13)

13. (a) Consider a boy who has a volleyball tournament the next day but feels sick today. Unusually, there is only a 40% chance he would fall sick since he is a healthy boy. Now, Find the probability of the boy participating in the tournament. The boy is very interested in volleyball, so there is a 90% probability that he will participate in tournaments and 20% fall sick, given that he participates in the tournament. (13)

Or

(b) Explicate the weighted K-nearest Neighbour algorithm with a suitable sketch. (13)

14. (a) Discuss the steps involved in the Backpropagation algorithm. (13)

Or

(b) Explain hyper-parameter tuning with example. (13)

15. (a) Mention the various methods of measuring Classifier Performance with suitable examples. (13)

Or

(b) List and illuminate the Guidelines for Machine Learning Experiments. (13)

PART C — (1 × 15 = 15 marks)

16. (a) Discuss the supervised and unsupervised learning with example.

Or

(b) What is KNN? Where are utilise the concept? List the importance with example.

Reg. No. : | E | N | G | G | T | R | E | E | . | C | O | M |

Question Paper Code : 30029

B.E./B.Tech. DEGREE EXAMINATIONS, APRIL/MAY 2023.

Fourth Semester

Artificial Intelligence and Data Science

AL 3451 — MACHINE LEARNING

(Common to Computer Science and Business Systems)

(Regulations 2021)

Time : Three hours | Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What do you mean by hypothesis space?
2. List few applications of Machine learning.
3. What is a gradient? How Gradient-Descent is useful in Machine learning?
4. Compare and contrast linear regression and logistic regression.
5. Distinguish between Bagging and Boosting.
6. Define voting.
7. List few activation functions.
8. What is a hyperparameter? List few hyperparameters.
9. Give the use of Mc Neman's test.
10. What is meant by Bootstrapping?

PART B — (5 × 13 = 65 marks)

11. (a) Elaborate on PAC Learning. Give an example.

Or

(b) (i) Discuss in detail about Bias-variance tradeoff. (6)

(ii) Explain about VC dimension. (7)

12. (a) Explain about how optimal Hyperplane differs from other Hyperplanes. Elaborate on how SVM is able to achieve this?

Or

(b) Explain the process of constructing CART (Classification And Regression Tree) with a suitable example.

13. (a) Explain about EM algorithm with suitable examples.

Or

(b) Explain the weighted K-nearest Neighbor algorithm with a suitable sketch.

14. (a) Discuss in detail about Backpropagation concept in ANN.

Or

(b) Elaborate on :
(i) Regularization
(ii) Drop-outs
(iii) Batch Normalization
(iv) Vanishing gradient

15. (a) (i) Elaborate on cross-validation approach. (8)
(ii) When do you decide on resampling? Explain. (5)

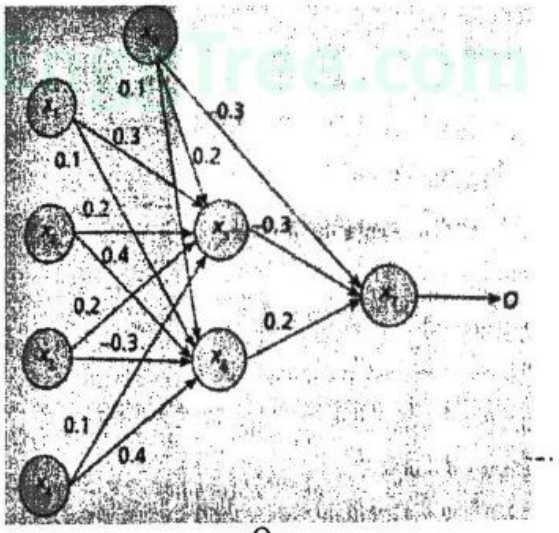
Or

(b) Represent and discuss the various methods of measuring Classifier performance.

PART C — (1 × 15 = 15 marks)

16. (a) Perform a feedforward operation in a Multi-Layer Perception and conclude the result. This given MLP consists of an Input, one hidden, and an Output layer. The input layer has 4 neurons, the hidden layer has 2 neurons, the output layer has a single neuron, and the Learning rate is 0.8.

X_1	X_2	X_3	X_4	$O_{Desired}$
1	1	0	1	1



Or

(b) Cluster the following data set using the k-means algorithm with an initial value of objects 2 and 5 with the coordinate values (4,6) and (12,4) as initial seeds.

OBJECTS	X-coordinate	Y-coordinate
1	2	4
2	4	6
3	6	8
4	10	4
5	12	4

Reg. No. : 4 2 1 6 2 2 2 4 3 0 3 3

Question Paper Code : 50070

B.E./B.Tech. DEGREE EXAMINATIONS, APRIL/MAY 2024.

Fourth Semester

Computer Science and Engineering (Artificial Intelligence and Machine Learning)

AL 3451 — MACHINE LEARNING

(Common to : Artificial Intelligence and Data Science/Computer Science and Business Systems)

(Regulations 2021)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Define Machine Learning.
2. Write some examples for machine learning applications.
3. Mention the merits of Bayesian linear regression.
4. Distinguish between Random Forest and Support Vector Machine.
5. When is supervised learning better than unsupervised learning?
6. Define Expectation Maximization.
7. Differentiate between Single layer and Multilayer Perceptron.
8. List the problems associated with Backpropagation Neural Network.
9. Recall the benefits of the Cross-Validation method.
10. How do you evaluate a Classification Algorithm?

PART B — (5 × 13 = 65 marks)

11. (a) Discuss the following

(i) Vapnik-Chervonenkis (VC) Dimension. (6)

(ii) Probably Approximately Correct (PAC) Learning. (7)

Or

(b) Write detailed notes on Inductive Bias and Bias variance trade-off.

12. (a) By the method of least squares find the straight line to the data given below.

X	5	10	15	20	25
Y	16	19	23	26	30

Or

(b) With an example explain the Decision Tree concepts in detail.

13. (a) Compare Bagging, Boosting and Stacking ensemble methods.

Or

(b) Cluster the following eight points (with (x, y) representing locations) into three clusters using K-means clustering method. A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9).

14. (a) Describe in brief about Multilayer perceptron activation functions.

Or

(b) Illustrate the following

(i) Batch Normalization. (7)

(ii) Dropout. (6)

15. (a) With neat diagram explain about K-fold Cross Validation technique.

Or

(b) Elaborate the t test, McNemar's test and K-fold CV paired t test by giving your own example.

PART C — (1 × 15 = 15 marks)

16. (a) Consider the following list that contains name, age, gender and class of sports. In the Gender field males are denoted by the numeric value 0 and females by 1. Using the K-Nearest Neighbor (KNN) algorithm, find class of sports for a girl whose name is Angelina, her k factor is 3, and her age is 5.

Ajay	32	0	Football
Mark	40	0	Neither
Sara	16	1	Cricket
Zaira	34	1	Cricket
Sachin	55	0	Neither
Rahul	40	0	Cricket
Pooja	20	1	Neither
Smith	15	0	Cricket
Laxmi	55	1	Football
Michael	15	0	Football

Or

(b) The grades of a class of 9 students on a midterm report (X) and on the final examination (Y) are as follows:

X	77	50	71	72	81	94	96	99	67
Y	82	66	78	34	47	85	99	99	68

(i) Estimate the linear regression line. (12)

(ii) Estimate the final examination grade of a student who received a grade of 85 on the midterm report. (3)

Reg. No. : | | | | | | | | | | | | |

Question Paper Code : 40075

B.E./B.Tech. DEGREE EXAMINATIONS, NOVEMBER/DECEMBER 2024.

Fourth Semester

Comp[illegible]nce and Engineering (Artificial Intelligence and Machine Learning

AL 3451 – MACHINE LEARNING

(Common to: Artificial Intelligence and Data Science/Computer Science and Business Systems)

(Regulations 2021)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Define generalisation in machine learning.
2. What is bias in machine learning?
3. Difference between regression and classification.
4. What is support vector in SVM?
5. Write the purpose of bagging in ensemble learning.
6. Define unsupervised learning.
7. List different activation function used in neural networks.
8. What is vanishing gradient problem?
9. Write the advantages of cross validation.
10. Define the concept of bootstrapping.

PART B — (5 × 13 = 65 marks)

11. (a) (i) Illustrate the trade off between the bias and variance in ML. (8)

(ii) Explain the concept of inductive Bias. (5)

Or

(b) Illustrate the concept of Vapnik-Chervonenkis (VC) dimension.

12. (a) (i) Discuss the model representation for logistic regression. (5)

(ii) Derive the Gradient of cost function (Error) for logistic regression which uses binary cross entropy. (8)

Or

(b) Illustrate the mathematical formulation for soft margin SVM and hard margin SVM.

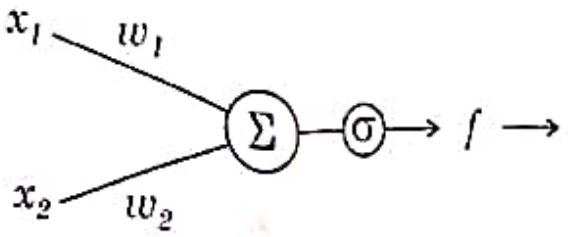
13. (a) Illustrate the principles of K-Means algorithms with an example.

Or

(b) (i) Explain the concept of Gaussian mixture model. (7)

(ii) Discuss in detail about Expectation Maximization (EM) algorithm. (6)

14. (a) Consider the following neural network problem, use back propagation algorithm to find the update of w_1.



$$f = \sum(x_1 * w_1 + x_2 * w_2)$$

Initial Value of $x_1 = 10$, $x_2 = 20$, $w_1 = 2$, $w_2 = 1$, alpha $(\sigma) = 0.2$. Actual Output = 0

Use Mean Square Error as loss function.

Or

(b) (i) Explain about the Back Propagation algorithm in Neural Network. (8)

(ii) Write short notes on Hyperparameter tunning. (5)

15. (a)

Cases	Actual Sick ?	Predicted Sick ?
1	1	1
2	1	0
3	0	1
4	0	0
5	1	1
6	0	0
7	1	0
8	0	1
9	1	1
10	0	0

Construct Confusion Matrix for the above table. Calculate Accuracy, Precision, Recall and F1 score. The values in the table denotes 1 for Yes and 0 for No.

Explain Why accuracy is generally not the preferred performance measure for classifiers, especially when you are dealing with skewed datasets (i.e., when some classes are much more frequent than others).

Or

(b) (i) Among 100 persons tested for cancer, actually(truly) 70 are affected by cancer. In the model prediction, 60 persons are predicted as cancer in which 15 are not affected by cancer (It means 15 persons are wrongly predicted as cancer). Construct Confusion Matrix and calculate Precision and Recall. (8)

(ii) Illustrate the need for Precision and Recall. (5)

PART C — (1 × 15 = 15 marks)

16. (a) Consider a function $f(x) = x2 + 2x + 3$, A data-scientist wishes to find the minimum of this function using gradient descent. He uses the following update rule:

$x(t+1) = x(t) - (2x(t) + 2)$

Given this information, answer questions (a) to (d).

(i) If the expression for the gradient is correct, what is the step-size that the data-scientist is using? (3)

(ii) If $x(0) = 0$, what is the value of $x(100)$? (4)

(iii) Justify why gradient descent update rule wont reach the minimum of the function $f(x)$? (4)

(iv) Using gradient descent with appropriate step size, find the value of x at which the function attains its minimum value, if such a minimum exists. (4)

Or

(b) Consider the following linear regression problem, use gradient decent to find the value of w_1 for two iteration.

$y = X * w_1 + b$

where y is the output of the model. X is input of the model

Initial Value of w_1=4, b = 5, alpha = 0.5.

Consider the Data points (5,14), (10,24), (15,32) for X and Y respectively.

Use Mean Square Error as loss function.