Question Bank

Units	Q.No	Question	Marks
UNIT-1 COMBINING DIFFERENT MODELS	1	Define Machine Learning.	1.5
	2	What is Model Selection in Machine Learning?	1.5
	3	Explain the importance of evaluating Machine Learning algorithms.	1.5
	4	What is the purpose of Cross-Validation in ML?	1.5
	5	Define Overfitting and Underfitting in Machine Learning.	1.5
	6	Explain the Bias-Variance Tradeoff.	2
	7	What is the difference between Training Error and Generalization Error?	2
	8	Define Ensemble Learning and give an example.	2
	9	What is Feature Importance in Random Forest?	2
	10	Explain the role of Hyperparameter Tuning in Machine Learning.	2
	11	Explain the concept of Bagging in Ensemble Learning with an example.	5
	12	Describe Boosting and how it improves model accuracy.	5
	13	Explain the working of Random Forest Algorithm.	5
	14	Discuss different Model Evaluation Metrics used in ML.	5
	15	Compare and contrast Bagging, Boosting, and Stacking.	5
Unit 2: Dimensionality Reduction	1	What is Dimensionality Reduction?	1.5
	2	Define Principal Component Analysis (PCA).	1.5
	3	What is Linear Discriminant Analysis (LDA)?	1.5
	4	Mention one key difference between PCA and LDA.	1.5
	5	What is the purpose of Factor Analysis?	1.5
	6	Explain how PCA helps in Dimensionality Reduction.	2
	7	What are eigenvalues and eigenvectors in PCA?	2
	8	How does Kernel PCA differ from traditional PCA?	2
	9	Explain the concept of Latent Variables in Factor Analysis.	2
	10	What is the significance of Independent Component Analysis (ICA) in signal processing?	2
	11	Describe the step-by-step working of Principal Component Analysis (PCA) with an example.	5
	12	Compare and contrast PCA, Kernel PCA, and Factor Analysis.	5
	13	Explain Factor Analysis and how it is used in Machine Learning.	5
	14	Discuss the importance of ICA in Machine Learning and provide a real-world application.	5
	15	How does Kernel PCA handle non-linear relationships in data? Explain with an example.	5
UNIT –III LEARNING WITH NEURAL NETWORKS	1	What is a Perceptron in Neural Networks?	1.5
	2	Define Deep Learning.	1.5
	3	What is the purpose of an Activation Function in Neural Networks?	1.5
	4	What is Feature Representation Learning?	1.5
	5	Mention one key difference between Artificial Neural Networks (ANN) and Deep Neural Networks (DNN).	1.5
	6	Explain the structure of a Multilayer Neural Network.	2
	7	What is the significance of weights and biases in Neural Networks?	2

	8	How does Backpropagation help in training Neural Networks?	2
	9	What is the difference between a Feedforward Neural Network and a Recurrent Neural Network (RNN)?	2
	10	Explain the role of Convolutional Neural Networks (CNNs) in Deep Learning.	2
	11	Describe the working of a Perceptron with an example.	5
	12	Explain how Deep Learning differs from traditional Machine Learning.	5
	13	Discuss the structure and learning process of a Multilayer Neural Network.	5
	14	What is Feature Representation Learning, and why is it important in Deep Learning?	5
	15	How do Neural Networks learn patterns from data? Explain with an example.	5
UNIT IV REINFORCEMENT LEARNING	1	What is Reinforcement Learning?	1.5
	2	List the key elements of Reinforcement Learning.	1.5
	3	What is the difference between Model-Based and Model-Free Reinforcement Learning?	1.5
	4	Define Policy in Reinforcement Learning.	1.5
	5	What is the Exploration-Exploitation Tradeoff in Reinforcement Learning?	1.5
	6	Explain how Reinforcement Learning differs from Supervised Learning.	2
	7	What is the role of Reward Function in Reinforcement Learning?	2
	8	Define Adaptive Dynamic Programming in the context of RL.	2
	9	What is Generalization in Reinforcement Learning?	2
	10	Explain how Policy Search is used in Reinforcement Learning.	2
	11	Describe the key elements of Reinforcement Learning with an example.	5
	12	Explain the concept of Policy Search and its role in Reinforcement Learning.	5
	13	What is Adaptive Dynamic Programming, and how is it applied in Reinforcement Learning?	5
	14	Discuss the role of Generalization in Reinforcement Learning and why it is important.	5
	15	How does Reinforcement Learning apply to real-world applications? Provide examples.	5

Unit -1 Solution

1. Define Machine Learning.
Machine Learning (ML) is a field of Artificial Intelligence (AI) that allows computers to learn from data and make decisions without being explicitly programmed. ML algorithms improve performance based on experience.

Example:

- Email spam filtering: ML models classify emails as **spam or not spam** based on historical data.

2. What is Model Selection in Machine Learning?
Model Selection is the process of choosing the best model from multiple candidates based on its performance on validation data.

Common Model Selection Techniques:

- **Cross-Validation**: Splitting data into training and validation sets.
- **Performance Metrics Comparison**: Choosing a model with high accuracy and generalization ability.

Example:
Selecting the best classification model between **Decision Trees, SVM, and Neural Networks** for spam detection.

3. Explain the importance of evaluating Machine Learning algorithms.
Evaluating ML models ensures that the algorithm is reliable, accurate, and generalizes well to unseen data.

Reasons for Model Evaluation:

- Measures **performance** using metrics like accuracy, precision, recall, F1-score.
- Identifies **overfitting** (memorizing data) and **underfitting** (too simplistic model).
- Helps in **model comparison** to select the best performing one.

Example:
A high training accuracy but poor test accuracy indicates **overfitting**, requiring improvements.

4. What is the purpose of Cross-Validation in ML?
Cross-validation is a technique to **assess model performance on different subsets of data** and avoid overfitting.

Types of Cross-Validation:

1. **K-Fold Cross-Validation**: Splits data into K equal parts, training on (K-1) parts and testing on the remaining.

2. **Leave-One-Out Cross-Validation (LOOCV)**: Uses one data point for testing while the rest are for training.

Example:
Using **5-Fold Cross-Validation** ensures every data point is used for both training and testing.

5. Define Overfitting and Underfitting in Machine Learning.

Feature	Overfitting	Underfitting
Definition	Model memorizes training data but fails on new data.	Model is too simple and fails to capture patterns.
Cause	Too complex model, too many features.	Model is too simple or lacks features.
Solution	Reduce complexity, use regularization (L1/L2), more training data.	Increase complexity, add more features.

Example:

- **Overfitting**: A deep decision tree that perfectly memorizes training data.
- **Underfitting**: A linear regression model used on non-linear data.

2 Marks Questions (Short Descriptive Answer)

6. Explain the Bias-Variance Tradeoff.
The Bias-Variance Tradeoff refers to the balance between **model complexity** and **generalization ability**.

- **High Bias (Underfitting)**: Model is too simple and fails to learn patterns.
- **High Variance (Overfitting)**: Model learns noise and performs well on training data but poorly on test data.

Solution:

- Use **Regularization** (L1/L2 penalties).
- Use **Ensemble Methods** (Bagging, Boosting).
- Use **Cross-Validation** to tune complexity.

7. What is the difference between Training Error and Generalization Error?

Feature	Training Error	Generalization Error
Definition	Error on training data.	Error on unseen test data.
Cause	Model complexity, incorrect assumptions.	Overfitting or underfitting.
Solution	Improve model architecture, better features.	Use cross-validation, increase data.

Example:
A neural network with **high training accuracy** but **low test accuracy** suffers from **high generalization error**.

8. Define Ensemble Learning and give an example.
Ensemble Learning is a technique that combines multiple models to improve overall accuracy.

Types of Ensemble Learning:

1. **Bagging (Bootstrap Aggregating)**: Reduces variance (Example: Random Forest).
2. **Boosting**: Reduces bias (Example: AdaBoost, Gradient Boosting).
3. **Stacking**: Combines diverse models with a meta-learner.

Example:

- **Random Forest**: Uses **multiple Decision Trees** to improve prediction accuracy.

9. What is Feature Importance in Random Forest?
Feature Importance in Random Forest helps determine which input variables are most influential in making predictions.

Methods to Determine Feature Importance:

1. **Gini Impurity**: Measures how often a feature reduces classification uncertainty.
2. **Permutation Importance**: Shuffles features and measures accuracy drop.

Example:

- In a **fraud detection model**, feature importance may rank **transaction amount** higher than **time of transaction**.

10. Explain the role of Hyperparameter Tuning in Machine Learning.
Hyperparameter tuning involves **optimizing model parameters** that are not learned during training.

Common Hyperparameters:

- **Learning Rate** (Neural Networks, Gradient Boosting).
- **Number of Trees** (Random Forest, XGBoost).
- **Regularization Strength** (L1, L2 penalties).

Methods for Hyperparameter Tuning:

1. **Grid Search**: Tests all parameter combinations.
2. **Random Search**: Randomly selects parameters.
3. **Bayesian Optimization**: Uses probability to find the best parameters efficiently.

Example:

- **Tuning the number of trees** in a **Random Forest** to improve accuracy without overfitting.

11. Explain the Concept of Bagging in Ensemble Learning with an Example.
What is Bagging?
Bagging (**Bootstrap Aggregating**) is an ensemble learning method that aims to improve the stability and accuracy of machine learning models by reducing variance and preventing overfitting. It works by creating multiple versions of the same model trained on different subsets of data and combining their predictions.

Working of Bagging:

1. **Data Bootstrapping:** Multiple subsets of data are created using **random sampling with replacement** from the original dataset.
2. **Independent Model Training:** Each subset is used to train a separate model of the same type (e.g., Decision Trees).
3. **Aggregation of Predictions:**
   - For **classification**, majority voting is used to determine the final class.
   - For **regression**, the predictions are averaged.

Example:
Random Forest is a classic example of bagging. It builds multiple decision trees using different subsets of data and averages their predictions. This reduces the likelihood of overfitting that is common with individual decision trees.

Advantages of Bagging:
Reduces variance and overfitting.
Works well with high-dimensional data.
Increases model stability and accuracy.

12. Describe Boosting and How It Improves Model Accuracy.
What is Boosting?
Boosting is an ensemble technique where models are trained **sequentially**, with each new model correcting the errors of its predecessors. It converts **weak learners** (models that perform slightly better than random guessing) into **strong learners**.

How Boosting Works:

1. A weak model is trained on the original dataset.
2. The errors (misclassified or high-error samples) from this model are identified and assigned **higher weights**.
3. A new model is trained, giving **more importance** to these misclassified instances.
4. The process is repeated until a stopping condition is met (e.g., a set number of iterations).
5. The final model combines the predictions of all weak models using a weighted sum.

Example: AdaBoost (Adaptive Boosting)

- The first decision tree (weak learner) is trained on the dataset.
- Misclassified points are assigned **higher weights**.
- The second tree focuses more on these misclassified points.
- This process continues until an optimal model is formed.

How Boosting Improves Accuracy:
Reduces bias by focusing on difficult examples.
Creates a **strong model** by iteratively refining weak models.
Works well with both classification and regression tasks.

Popular Boosting Algorithms:

- **AdaBoost (Adaptive Boosting)**
- **Gradient Boosting**
- **XGBoost (Extreme Gradient Boosting)** – Optimized for speed and performance.

13. Explain the Working of Random Forest Algorithm.
What is Random Forest?
Random Forest is an ensemble learning algorithm based on **Bagging**, where multiple decision trees are built using randomly selected subsets of data and features. The final output is determined by aggregating their predictions.

Working of Random Forest:

1. **Data Sampling:** The dataset is divided into multiple **bootstrap** samples (randomly sampled subsets with replacement).
2. **Feature Selection:** Each decision tree selects a random subset of features at each split to introduce randomness.
3. **Tree Training:** Multiple decision trees are trained independently on different data subsets.
4. **Prediction Aggregation:**
    - **For classification**: Uses **majority voting** among the trees.
    - **For regression**: Uses **average prediction** across trees.

Example:
A Random Forest model trained to predict whether an email is spam or not will generate multiple decision trees, each using different subsets of emails and features. The final classification is based on the majority vote from all decision trees.

Advantages of Random Forest:
Reduces overfitting compared to a single decision tree.
Works well with large datasets.
Handles missing data and noisy inputs effectively.

14. Discuss Different Model Evaluation Metrics Used in ML. (5 Marks)
Model evaluation metrics help assess the **performance** of machine learning models. The choice of metric depends on whether the task is **classification** or **regression**.

1. Classification Metrics:
Used when the output is categorical (e.g., spam/not spam, disease/no disease).

- **Accuracy:**

  Accuracy=TP+TNTP+TN+FP+FN

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

Accuracy=TP+TN+FP+FNTP+TN

Measures the percentage of correct predictions.

- **Precision:**

  Precision=TPTP+FPPrecision = \frac{TP}{TP + FP}Precision=TP+FPTP

  Measures how many **predicted positive cases** are actually positive.

- **Recall (Sensitivity):**

  Recall=TPTP+FNRecall = \frac{TP}{TP + FN}Recall=TP+FNTP

  Measures how many **actual positive cases** were predicted correctly.

- **F1 Score:**

  F1=2×Precision×RecallPrecision+RecallF1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}F1=2×Precision+RecallPrecision×Recall

  Harmonic mean of precision and recall, used when there is an imbalance in classes.

- **ROC-AUC Score (Receiver Operating Characteristic - Area Under Curve):**
  Measures the trade-off between **True Positive Rate (Recall)** and **False Positive Rate**.

2. Regression Metrics:
Used when the output is continuous (e.g., predicting house prices).

- **Mean Absolute Error (MAE):**

  MAE=1n∑|yactual−ypredicted|MAE = \frac{1}{n} \sum |y_{actual} - y_{predicted}|MAE=n1∑|yactual−ypredicted|

  Measures the average absolute difference between actual and predicted values.

- **Mean Squared Error (MSE):**

  MSE=1n∑(yactual−ypredicted)2MSE = \frac{1}{n} \sum (y_{actual} - y_{predicted})^2MSE=n1∑(yactual−ypredicted)2

  Penalizes larger errors more than MAE.

- **R^2 Score (Coefficient of Determination):**
  Measures how well the model explains variance in the data.

15. Compare and Contrast Bagging, Boosting, and Stacking.

Feature	Bagging	Boosting	Stacking
Concept	Models trained in parallel	Models trained sequentially	Combines multiple models at different levels
Goal	Reduce variance	Reduce bias and improve accuracy	Improve generalization
Dependency	Independent models	Each model depends on the previous one	Uses multiple diverse models
Handling of Errors	Reduces overfitting by averaging	Focuses on misclassified samples	Uses a meta-model to optimize predictions
Example Algorithm	Random Forest	AdaBoost, XGBoost	Stacked Generalization

Key Takeaways:

- **Bagging**: Trains models in **parallel** to reduce variance and improve stability. Example: **Random Forest**.
- **Boosting**: Trains models **sequentially**, focusing on errors to reduce bias. Example: **XGBoost**.
- **Stacking**: Combines multiple models and trains a **meta-model** to make final predictions. Example: **Stacked Generalization**.

Unit -2 Solution

1. What is Dimensionality Reduction?

Dimensionality Reduction is a technique used in machine learning to reduce the number of input features (variables) in a dataset while preserving essential information. High-dimensional data can lead to:

Increased computational complexity.
Overfitting due to redundant or irrelevant features.
Difficulty in data visualization.

Types of Dimensionality Reduction:

- **Feature Selection:** Selecting a subset of important features.
- **Feature Extraction:** Transforming data into a lower-dimensional space (e.g., PCA, LDA).

2. Define Principal Component Analysis (PCA).

Principal Component Analysis (PCA) is an **unsupervised** dimensionality reduction technique that converts a high-dimensional dataset into a new set of **principal components**, which are uncorrelated and ordered by importance (variance).

How PCA Works:

1. Standardize the data.
2. Compute the covariance matrix.

3. Find eigenvalues and eigenvectors.
4. Select the top **k** eigenvectors (principal components).
5. Project the data onto the new lower-dimensional space.

Application: Used in image compression, facial recognition, and feature reduction in ML models.

3. What is Linear Discriminant Analysis (LDA)?
Linear Discriminant Analysis (LDA) is a **supervised** dimensionality reduction technique that maximizes class separability. It finds a linear combination of features that best separates multiple classes.

How LDA Works:

1. Compute the **scatter matrices** (within-class and between-class scatter).
2. Find the eigenvalues and eigenvectors.
3. Select the top **k** discriminant components.
4. Project the data onto the new space for better class separation.

Application: Used in face recognition, medical diagnostics, and classification tasks.

4. Mention One Key Difference Between PCA and LDA.

Feature	PCA	LDA
Type	Unsupervised	Supervised
Goal	Maximizes variance	Maximizes class separation
Use Case	Works with any data	Used for classification problems
Data Dependency	Works with unlabeled data	Requires labeled data

Key Difference: PCA **does not consider class labels**, while LDA **requires class labels** to maximize class separability.

5. What is the Purpose of Factor Analysis?
Factor Analysis is a **statistical method** used to identify underlying **latent variables** that explain patterns in the observed data.

Purpose of Factor Analysis:

- Reduces dimensionality by grouping correlated features.
- Identifies hidden patterns in datasets.
- Helps in psychology, finance, and market research to uncover hidden relationships.

Example: In personality tests, multiple questions may measure an **underlying trait** like **introversion or extroversion**.

6. Explain How PCA Helps in Dimensionality Reduction.
PCA reduces dimensionality by finding the **principal components** that retain the most variance while discarding less informative features.

How PCA Reduces Dimensionality:

1. **Find Correlation Between Features:** Identifies redundant information.

2. **Compute Principal Components:** New axes that maximize variance.
3. **Select Top Components:** Keeping only the most informative ones.
4. **Transform Data:** Project data onto the new lower-dimensional space.

Example: If we have **100 features**, PCA might reduce them to **10 principal components** while keeping **95% of the variance**.

Benefit: PCA speeds up computation and improves model performance while preventing overfitting.

7. What Are Eigenvalues and Eigenvectors in PCA?
Eigenvalues and eigenvectors are fundamental to PCA and help in finding the principal components.

Definitions:

- **Eigenvectors**: Define the direction of the principal components.
- **Eigenvalues**: Measure the amount of variance captured by each eigenvector.

How They Work in PCA:

1. Compute the **covariance matrix** of the data.
2. Find the **eigenvalues and eigenvectors**.
3. Select eigenvectors with the **highest eigenvalues** to form principal components.

Example: In PCA, the first principal component has the **highest eigenvalue**, capturing the most variance.

Interpretation: Higher eigenvalues mean **more important components**.

8. How Does Kernel PCA Differ from Traditional PCA?
Traditional PCA is **linear**, meaning it can't capture complex relationships. **Kernel PCA** extends PCA to handle **non-linear** data using the **kernel trick**.

Key Differences:

Feature	Traditional PCA	Kernel PCA
Type	Linear	Non-linear
Data Assumption	Assumes linearity	Can capture non-linear relationships
Kernel Trick	Not used	Used to map data into a higher-dimensional space

Example: Kernel PCA is useful in **image recognition** where data is often non-linearly distributed.

9. Explain the Concept of Latent Variables in Factor Analysis.
Latent Variables are **hidden variables** that influence observed data but are not directly measured.

How Latent Variables Work in Factor Analysis:

- Factor Analysis assumes that **multiple observed variables** can be explained by a **smaller set of latent variables**.
- These latent factors help in reducing dimensionality and understanding patterns in data.

Example: In intelligence testing, different test scores (math, verbal, reasoning) may be influenced by a latent factor called **general intelligence (IQ)**.

Benefit: Helps in feature reduction and pattern discovery.

10. What is the Significance of Independent Component Analysis (ICA) in Signal Processing?

Independent Component Analysis (ICA) is used in **signal processing** to separate **mixed signals** into their independent sources.

Significance of ICA in Signal Processing:
Extracts meaningful **independent components** from mixed signals.
Helps in **blind source separation** (BSS) to separate overlapping signals.
Works well with **non-Gaussian** and complex datasets.

Applications:

- **Audio Processing:** Separating multiple voices recorded together (e.g., "Cocktail Party Problem").
- **EEG & ECG Signal Analysis:** Removing noise from brain and heart signals.

Difference from PCA: ICA looks for **statistical independence**, whereas PCA looks for **uncorrelated components**.

11. Describe the Step-by-Step Working of Principal Component Analysis (PCA) with an Example.
Principal Component Analysis (PCA) is a **dimensionality reduction** technique that transforms high-dimensional data into a lower-dimensional space while preserving maximum variance.

Step-by-Step Working of PCA:
1 **Standardize the Data**

- Convert all features to have **zero mean** and **unit variance** to avoid bias due to different scales.
- Formula for standardization:

  Xscaled=X−μσX_{\text{scaled}} = \frac{X - \mu}{\sigma}Xscaled =σX−μ

where μ\muμ is the mean and σ\sigmaσ is the standard deviation.

2 **Compute the Covariance Matrix**

- The covariance matrix shows the relationship between different features.
- If a high covariance exists between two features, they carry redundant information.
- Covariance formula:

  Cov(X,Y)=1n∑(Xi−X¯)(Yi−Y¯)\text{Cov}(X, Y) = \frac{1}{n} \sum (X_i - \bar{X})(Y_i - \bar{Y})Cov(X,Y)=n1∑(Xi−X¯)(Yi−Y¯)

3 **Find Eigenvalues and Eigenvectors**

- Eigenvalues represent the **amount of variance** captured by each principal component.
- Eigenvectors define the **direction** of new feature axes.

- Solve:

  Ax=λxAx = \lambda xAx=λx

  where AAA is the covariance matrix, λ\lambdaλ are eigenvalues, and xxx are eigenvectors.

4 **Select Principal Components**

- Choose the top **k** eigenvectors corresponding to the **largest eigenvalues**.
- These eigenvectors form the **principal components** (new feature axes).

5 **Project Data onto the New Space**

- Transform original data into the new lower-dimensional space using:

  X'=XWX' = X WX'=XW

  where WWW is the matrix of selected eigenvectors.

Example of PCA in Image Compression:

- Consider **grayscale images of handwritten digits (28×28 pixels = 784 features)**.
- PCA can **reduce dimensions** from **784 to 100 features**, retaining most of the information.
- The compressed images can be used for faster digit recognition.

Key Benefit: PCA reduces dimensions, **removes redundancy**, and speeds up computations.

12. Compare and Contrast PCA, Kernel PCA, and Factor Analysis.

Feature	**PCA**	**Kernel PCA**	**Factor Analysis**
Type	Linear	Non-linear	Statistical
Goal	Maximizes variance	Captures non-linear patterns	Identifies latent factors
Assumption	Data is linearly related	Uses kernel trick for non-linearity	Factors explain correlations
Mathematical Basis	Eigenvalues and eigenvectors	Kernel functions (RBF, polynomial)	Covariance and latent variables
Application	Feature extraction, noise reduction	Non-linear classification, clustering	Psychology, social sciences, finance

Key Difference:

- PCA assumes a **linear** relationship.
- Kernel PCA can **capture non-linear relationships**.
- Factor Analysis is used for identifying **hidden underlying factors**.

13. Explain Factor Analysis and How It Is Used in Machine Learning.
Factor Analysis (FA) is a **statistical technique** used to uncover **latent (hidden) variables** that explain correlations between observed variables.

Key Concepts of Factor Analysis:

- **Observed Variables:** The features present in the dataset.
- **Latent Factors:** Underlying variables that influence multiple observed features.
- **Factor Loadings:** Show how strongly each observed variable is related to a latent factor.

Steps in Factor Analysis:
1 **Compute the Correlation Matrix**: Determines relationships between variables.
2 **Extract Factors**: Identify groups of correlated features.
3 **Rotation of Factors**: Simplify interpretation (e.g., Varimax rotation).
4 **Interpret Factors**: Assign meaning to discovered factors.

How Factor Analysis is Used in ML:
Dimensionality Reduction: Reduces feature space while maintaining meaningful patterns.
Psychological Testing: Identifies personality traits from survey data.
Market Research: Understands consumer preferences by grouping similar responses.
Finance: Analyzes stock market movements by identifying common trends among stocks.

Example: In a personality test, multiple questions measure an **underlying personality trait** like extroversion. Factor Analysis can reveal that **certain questions are correlated** and represent the same hidden factor.

14. Discuss the Importance of ICA in Machine Learning and Provide a Real-World Application.
Importance of Independent Component Analysis (ICA) in ML:
ICA is used to extract **independent signals** from mixed data. Unlike PCA, which focuses on **uncorrelated** components, ICA finds components that are **statistically independent**.

Key Features of ICA:

- Does not assume Gaussian distribution.
- Works well for separating **overlapping signals**.
- Useful in **unsupervised learning** and **blind source separation**.

Real-World Application: EEG Signal Processing
Problem: EEG (electroencephalogram) signals are recorded from multiple electrodes, often containing noise (e.g., eye movement, muscle activity).
Solution:

- ICA separates the **pure brain signal** from noise.
- Helps in detecting brain diseases like **epilepsy** and **Alzheimer's**.

Other Applications:

Speech Processing: Separating multiple speakers in a noisy room (Cocktail Party Problem).
Financial Data Analysis: Extracting independent trends in stock prices.
Image Processing: Removing unwanted noise from images.

Difference from PCA: PCA finds uncorrelated components, while ICA finds **statistically independent** ones.

15. How Does Kernel PCA Handle Non-Linear Relationships in Data? Explain with an Example.
Traditional PCA assumes **linear** relationships between features. **Kernel PCA** extends PCA to **non-linear** data using the **kernel trick**.

How Kernel PCA Works:
1 **Map Data into Higher-Dimensional Space**

- Uses kernel functions to transform data into a higher-dimensional space where it becomes **linearly separable**.

2 **Compute Kernel Matrix**

- Instead of using the covariance matrix, Kernel PCA calculates a kernel matrix to find principal components in the transformed space.

3 **Extract Principal Components**

- Apply eigenvalue decomposition to find the most significant non-linear patterns.

Example: Spiral Dataset Classification
Problem: Consider a dataset where points are arranged in **two interleaving spirals**. Traditional PCA fails because the data is **non-linearly distributed**.
Solution: Kernel PCA with an **RBF kernel** maps the data into a **higher-dimensional space**, making it linearly separable.

Before Kernel PCA: The dataset appears entangled.
After Kernel PCA: The transformed dataset becomes **linearly separable** by a simple classifier.

Common Kernel Functions in Kernel PCA:

Kernel Type	Formula	Use Case
Linear Kernel	K(x,y)=xTyK(x,y) = x^T yK(x,y)=xTy	When data is already linear
Polynomial Kernel	K(x,y)=(xTy+c)dK(x,y) = (x^T y + c)^dK(x,y)=(xTy+c)d	Captures curved patterns
RBF Kernel	(K(x,y) = e^{-\frac{	x-y

Unit-3

1. What is a Perceptron in Neural Networks?

A **Perceptron** is the most basic neural network model that acts as a **binary classifier**. It takes multiple inputs, applies weights, and passes the sum through an **activation function** to produce an output.

Structure of a Perceptron:
Inputs (x1,x2,...,xnx_1, x_2, ..., x_nx1,x2,...,xn) – Feature values.
Weights (w1,w2,...,wnw_1, w_2, ..., w_nw1,w2,...,wn) – Determines feature importance.
Bias (bbb) – Helps adjust the output.
Activation Function – A threshold function to decide the final output.

Mathematical Representation:
y=f(∑wixi+b)y = f\left(\sum w_i x_i + b\right)y=f(∑wixi+b)
where f(x)f(x)f(x) is an activation function.

Example: AND Gate
If inputs x1,x2x_1, x_2x1,x2 are binary (0 or 1), then:

- Output is 1 **only if both inputs are 1**.
- Otherwise, the output is 0.

Limitation: Perceptrons can only solve **linearly separable problems**.

2. Define Deep Learning.

Deep Learning is a subfield of **Machine Learning** that utilizes **deep neural networks** (DNNs) with multiple layers to automatically learn features from data.

Based on **Artificial Neural Networks (ANNs)**.
Uses **large datasets** and **high computational power**.
Enables tasks like **speech recognition, image classification, and NLP**.

Example:

- **Traditional ML:** Requires manual feature extraction (e.g., edge detection in images).
- **Deep Learning:** Learns features automatically using CNNs.

3. What is the Purpose of an Activation Function in Neural Networks?

Activation functions introduce **non-linearity** into a neural network, allowing it to learn complex relationships.

Common Activation Functions:
ReLU (Rectified Linear Unit):

f(x)=max (0,x)f(x) = \max(0, x)f(x)=max(0,x)

Sigmoid Function:

f(x)=11+e−xf(x) = \frac{1}{1+e^{-x}}f(x)=1+e−x1
Converts input into a probability between **0 and 1**.

Tanh (Hyperbolic Tangent):

f(x)=ex−e−xex+e−xf(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}f(x)=ex+e−xex−e−x
Output ranges between **-1 to 1**, better than Sigmoid.

4. What is Feature Representation Learning?

Feature Representation Learning is the ability of neural networks to automatically extract relevant features from raw data.

Reduces the need for **manual feature engineering**.
Used in **CNNs for image features** and **RNNs for time series data**.
Helps models achieve **higher accuracy**.

Example:

- **Traditional ML:** Needs domain expertise to extract features.
- **Deep Learning:** Learns **edges, shapes, and textures** automatically.

5. Key Difference Between ANN and DNN.

Aspect	ANN (Shallow Network)	DNN (Deep Network)
Number of Layers	1-2 layers	More than 3 layers
Feature Learning	Manual	Automatic
Performance	Limited	Excellent for complex tasks

Key Takeaway: DNNs perform better because they learn hierarchical features.

6. Structure of a Multilayer Neural Network.

A Multilayer Neural Network consists of:

Input Layer – Accepts input features.
Hidden Layers – Processes data using weighted connections.
Output Layer – Produces final predictions.

Why Use Multiple Layers?

- Can learn **complex patterns** in data.
- Each layer captures **different levels of abstraction**.

7. Significance of Weights and Biases in Neural Networks.
Weights (W): Control the strength of connections between neurons.
Bias (B): Helps shift the activation function for better learning.

Impact:

- Incorrect weights lead to **poor learning**.
- Bias prevents neurons from being **too restrictive**.

8. How Does Backpropagation Help in Training Neural Networks? (2 Marks)
Backpropagation is the algorithm used to **adjust weights and biases** to minimize error.

Steps in Backpropagation:
1 **Forward Pass:** Compute the output.
2 **Compute Loss:** Compare output with actual value.
3 **Backward Pass:** Compute gradients using **Chain Rule**.
4 **Weight Update:** Adjust weights using Gradient Descent.

Equation:

W=W−η∂L∂WW = W - \eta \frac{\partial L}{\partial W}W=W−η∂W∂L
where **η** = learning rate.

9. Difference Between Feedforward and Recurrent Neural Networks (RNN).

Feature	Feedforward NN	RNN
Data Flow	One direction	Loops present
Memory	No memory	Stores past states
Use Case	Image classification	Time series, speech processing

Key Difference: RNNs **retain memory**, making them useful for sequential data.

10. Role of CNNs in Deep Learning.
CNNs are specialized for **image processing** and **pattern recognition**.

Uses **Convolutional Layers** to detect edges, textures, and shapes.
Includes **Pooling Layers** to reduce dimensionality.

Example: Used in **self-driving cars, medical image analysis, and facial recognition**.

11. Working of a Perceptron with an Example.
A Perceptron **classifies data into two categories** using a decision boundary.

Example: AND Gate

- If both inputs are 1, output is 1.
- Otherwise, output is 0.

Truth Table:

x1	x2	Output
0	0	0
0	1	0
1	0	0
1	1	1

12. How Deep Learning Differs from Traditional ML.

Feature	Traditional ML	Deep Learning
Feature Engineering	Manual	Automatic
Data Requirement	Less	Large datasets needed
Performance	Good for structured data	Best for unstructured data

Deep Learning is powerful but requires large datasets.

13. Structure & Learning Process of Multilayer Neural Networks.

Input Layer – Takes raw data.
Hidden Layers – Learn patterns.
Output Layer – Produces final decision.

Learning Process:
1 Forward Propagation
2 Compute Loss
3 Backpropagation
4 Update Weights

14. Importance of Feature Representation Learning in Deep Learning

Feature Representation Learning allows **models to learn the best feature representations**.

Removes need for **manual feature selection**.
Used in **CNNs for images** and **LSTMs for text**.

Example: In self-driving cars, deep learning models extract **road features automatically**.

15. How Do Neural Networks Learn Patterns?

Adjusts weights based on training data.
Uses **Gradient Descent** to minimize loss.

Example: Handwritten digit recognition.
1 Train on labeled images.
2Use **backpropagation** to improve accuracy.
3 Predicts digits for new images.

Unit-4

1. What is Reinforcement Learning?

Reinforcement Learning (RL) is a type of **machine learning** where an **agent** learns to make decisions by interacting with an **environment**. The agent receives **rewards or penalties** based on its actions and improves its decision-making over time.

Key Features:
Trial and Error Learning – Learns by trying different actions.
Reward-Based Feedback – Gets a positive or negative reward.
Goal-Oriented – Maximizes cumulative reward over time.

Example:

- In a **chess game**, an RL agent learns the best moves by playing multiple games and adjusting its strategy based on wins/losses.

2. Key Elements of Reinforcement Learning.

Reinforcement Learning consists of the following components:

Element	Description
Agent	The decision-maker (e.g., robot, AI).
Environment	The world where the agent interacts.
State (S)	A specific situation in the environment.
Action (A)	The decision taken by the agent.
Reward (R)	Feedback signal (positive or negative).
Policy (π)	Strategy used by the agent to decide actions.
Value Function (V)	Measures how good a state is.
Model (Optional)	Predicts future states based on actions.

Example:

- **Self-driving cars** use RL to learn how to drive efficiently by interacting with roads and receiving rewards for safe driving.
-

3. Difference Between Model-Based and Model-Free RL.

Feature	Model-Based RL	Model-Free RL
Definition	Learns a model of the environment before decision-making.	Learns by direct interaction without a model.
Computation	Requires more memory and computation.	Less computationally expensive.
Learning Speed	Faster learning if model is accurate.	Slower but flexible.
Example	Chess AI (predicts opponent's moves).	Q-Learning in games.

Example:

- **Model-Based RL:** Predicting customer behavior in an e-commerce website before making recommendations.
- **Model-Free RL:** Learning to play a game by directly interacting with it.

4. Define Policy in Reinforcement Learning

A **policy (π)** defines the strategy used by an agent to select actions.

Deterministic Policy (π(s)) → Always selects the same action for a given state.
Stochastic Policy (π(a|s)) → Selects actions based on probability.

Example:

- In **tic-tac-toe**, a policy determines whether to play in the center or corner first.

5. What is the Exploration-Exploitation Tradeoff?

The **Exploration-Exploitation Tradeoff** is a key challenge in RL:

Exploration: Trying new actions to discover better rewards.
Exploitation: Using known actions to maximize reward.

Example:

- In **online advertising**, an RL system must decide whether to show a new ad (explore) or show a proven ad (exploit).

Solution:
Use **ε-greedy strategy**, where the agent explores with a small probability (ε) and exploits otherwise.

6. Difference Between Reinforcement Learning and Supervised Learning.

Aspect	Reinforcement Learning (RL)	Supervised Learning
Training Data	Learns from rewards via trial and error.	Learns from labeled data.
Feedback	Delayed feedback (rewards over time).	Immediate feedback (right or wrong).

Goal	Maximizing long-term reward.	Minimizing prediction error.
Example	**Chess AI** learning optimal moves.	**Image classifier** predicting dog vs. cat.

Key Difference: RL **learns dynamically**, while supervised learning relies on pre-existing data.

7. Role of Reward Function in Reinforcement Learning.

The **reward function** in RL determines the immediate feedback an agent gets after taking an action.

Purpose:

Encourages the agent to learn good actions.
Helps in **optimizing long-term rewards**.

Example:

- In **robot navigation**, the robot gets a reward for reaching its destination and a penalty for hitting obstacles.

Challenge:

Designing a good reward function is **critical** to achieving desired behavior.

8. Define Adaptive Dynamic Programming (ADP) in RL.

Adaptive Dynamic Programming (ADP) is a **Model-Based RL technique** that learns and improves policies by approximating:

Value Function (V) → Measures the goodness of states.
Policy (π) → Determines the best action.

Steps:

1 **Model Learning:** Estimates state transitions and rewards.
2**Policy Improvement:** Updates policy using learned model.
3 **Value Iteration:** Optimizes long-term rewards.

Example:

- **Traffic light optimization** using ADP to minimize congestion over time.

9. What is Generalization in Reinforcement Learning?

Generalization in RL means the ability of an agent to perform well on **unseen states** or **environments**.

Why Important?

RL should not memorize actions for every state.
It should learn general strategies.

Example:

- A **self-driving car** should be able to drive on new roads without explicit training.

Solution:

- Use **function approximation** (e.g., Deep Q-Networks) instead of storing values for each state.

10. How is Policy Search Used in RL?
Policy Search refers to optimization techniques used to find the best policy (π) in RL.

Types of Policy Search:
Gradient-Based Search: Uses policy gradients (e.g., REINFORCE algorithm).
Evolutionary Algorithms: Uses genetic algorithms to evolve better policies.

Example:

- **Robot Locomotion:** Policy search is used to teach robots how to walk efficiently.

Key Takeaway: Policy search is crucial in **continuous action spaces**, where traditional Q-learning fails.

11. Key Elements of Reinforcement Learning
Reinforcement Learning (RL) consists of the following **key elements**:

Elements of RL
1**Agent** → The learner/decision-maker.
2 **Environment** → The world where the agent operates.
3 **State (S)** → The current situation in the environment.
4 **Action (A)** → The choices available to the agent.
5 **Reward (R)** → Feedback received for an action.
6 **Policy (π)** → The strategy used to choose actions.
7 **Value Function (V)** → Expected long-term reward of a state.
8 **Model (Optional)** → Simulates the environment.

Example: RL in a Self-Driving Car
- **Agent** → The self-driving car.
- **Environment** → The road, traffic signals, and pedestrians.
- **State (S)** → Current speed, location, and traffic conditions.
- **Action (A)** → Accelerate, brake, steer left/right.
- **Reward (R)** → +10 for safe driving, -20 for breaking traffic rules.
- **Policy (π)** → Choose the safest driving action.
- **Value Function (V)** → Predicts safety and efficiency of future states.

Receives Reward (R) and New State (S)
Real-world Application: Google's **DeepMind AlphaGo** used RL to master the game of Go.

12. Policy Search in Reinforcement Learning
What is Policy Search?
Policy search refers to **finding the best policy (π)** that maximizes long-term rewards in RL.

Policy (π): A function that maps states to actions.
Goal: Optimize the policy for better decision-making.

Types of Policy Search
1 **Gradient-Based Methods** → Uses policy gradients to improve decisions (e.g., REINFORCE algorithm).
2 **Evolutionary Algorithms** → Uses genetic evolution to evolve better policies.
3 **Black-Box Optimization** → Uses optimization algorithms like CMA-ES.

Example: RL in Robotics
A robot arm learns how to pick up objects using **policy search**:

- **Initial Policy:** Randomly picks up objects.
- **Learning Process:** Improves its grip through **trial and error**.
- **Final Policy:** Learns the optimal way to pick up objects.

Real-world Application: Policy search is used in **robotic grasping**.

13. Adaptive Dynamic Programming (ADP) in RL
What is Adaptive Dynamic Programming?
Adaptive Dynamic Programming (ADP) is a **model-based RL technique** where the agent learns an internal model of the environment.

Steps in ADP:
1 **Model Learning** → Learns state transitions and rewards.
2 **Policy Iteration** → Updates the policy using the learned model.
3 **Value Iteration** → Optimizes rewards over time.

Example: Traffic Light Optimization
Goal: Minimize traffic congestion.

- The **traffic system (Agent)** observes congestion (State).
- It adjusts **traffic signals (Action)**.
- If congestion reduces, it gets a **positive reward**.
- The policy improves over time.

Real-world Application: Smart traffic control in cities uses ADP.

14. Role of Generalization in RL
What is Generalization in RL?
Generalization means that the RL agent should perform well on **new and unseen states**.

Why is Generalization Important?
Prevents **overfitting** to specific situations.
Helps RL models **adapt** to real-world scenarios.
Allows models to work **efficiently on large environments**.

Example: Self-Driving Car
A car trained only on **one city** must generalize to **other cities**:

- If it only memorizes specific roads, it fails in new areas.
- A **generalized policy** lets it drive safely anywhere.

Real-world Application: Tesla's Autopilot generalizes across multiple driving conditions.

15. Real-World Applications of Reinforcement Learning (5 Marks)
Reinforcement Learning is widely used in **real-world applications**:

Application	Use Case
Self-Driving Cars	Tesla's autopilot learns to navigate roads.
Healthcare	AI optimizes treatment plans.
Finance	RL-driven stock market trading.
Robotics	Robots learn to walk and perform tasks.
Gaming	AlphaGo and OpenAI Five master games.

Example 1: Stock Trading
- **Agent:** AI stock trader.
- **State:** Current market prices.
- **Action:** Buy, sell, or hold.
- **Reward:** Profit/loss based on stock movement.
  Used by: Hedge funds and investment firms.

Example 2: AI in Video Games
RL is used in **AI opponents in games**:

- **DeepMind's AlphaGo** defeated human Go players.
- **OpenAI Five** learned to play **Dota 2** at a world-class level.