MACHINE LEARNING

UNIT-II:

DECISION TREE LEARNING - Decision tree learning algorithm-Inductive bias- Issues in Decision tree learning;

ARTIFICIAL NEURAL NETWORKS – Perceptrons, Gradient descent and the Delta rule, Adaline, Multilayer networks, Derivation of backpropagation rule Backpropagation AlgorithmConvergence, Generalization;

PYQ

7marks:-

1.What are the type of problems in which Artificial Neural Network can be applied?

2.Derive the Backpropagation rule considering the training rule for Output Unit weights and Training Rule for Hidden Unit weights?

3. Relate Inductive bias with respect to Decision tree learning.

4. Define Neural Network? Explain Feed forward Neural Network.

5. Compare and contrast single layered model and multilayer perceptron model.

6. Define decision tree? Explain the function of decision tree.

2marks:-

1. Differentiate between Training data and Testing Data.

2. What are the issues in Machine Learning.

3. Explain perceptron in Artificial Neural Networks?

4. Describe the role of bias in ANN?

5. List the issues in Decision Tree Learning.

6. Explain gradient descent delta rule?

DECISION TREE

A decision tree is a graphical representation of a decision-making process. It's a tree-like model that illustrates the possible outcomes of a series of related choices. Decision trees are commonly used in machine learning, data analysis, and business decision-making.

Structure of a Decision Tree

1. **Root Node**: Represents the entire dataset and the initial decision to be made.
2. **Internal Nodes**: Represent decisions or tests on attributes. Each internal node has one or more branches.
3. **Branches**: Represent the outcome of a decision or test, leading to another node.
4. **Leaf Nodes**: Represent the final decision or prediction. No further splits occur at these nodes.

Function of a Decision Tree:

1.**Classify data/Classification**: Decision trees are used to classify data into different categories based on a set of input features.

2.**Predict outcomes/Regression**: By traversing the tree, you can predict the outcome of a decision based on the input features.

3.**Identify relationships**: Decision trees help identify relationships between input features and the target variable.

4. **Feature Selection:** The decision tree selects the most significant features to split the data. This selection is often based on criteria like Gini impurity, entropy (information gain), or variance reduction.

Issues in Decision Tree Learning.

- **Overfitting**:
    - Decision trees can become too complex, capturing noise in the training data. This leads to overfitting, where the model performs well on training data but poorly on unseen data.

- **Underfitting**:
  - Conversely, overly simple trees may underfit the data, failing to capture important patterns. This results in poor performance on both training and test datasets.
- **Sensitivity to Small Changes in Data**:
  - Small changes in the training data can lead to significantly different trees being generated.
- **Complexity with Large Datasets**:
  - As the size of the dataset grows, decision trees can become large and difficult to interpret. They may also require significant computational resources to construct and prune.
- **Lack of Robustness**:
  - Decision trees can be sensitive to outliers, which can distort the tree structure and decision-making process.
- **Interpretability vs. Accuracy Trade-off**:
  - Simpler trees are easier to interpret but may not capture complex patterns, while more complex trees may be accurate but hard to interpret.
- **Pruning Challenges**:
  - Deciding when and how to prune a decision tree to prevent overfitting while maintaining accuracy can be challenging and may require careful tuning

Pruning

To overcome **overfitting, pruning** techniques are used. Pruning reduces the size of the tree by removing nodes that provide little power in classifying instances. There are two main types of pruning:

- **Pre-pruning (Early Stopping)**: Stops the tree from growing once it meets certain criteria (e.g., maximum depth, minimum number of samples per leaf).
- **Post-pruning**: Removes branches from a fully grown tree that do not provide significant power.

Applications of Decision Trees

- **Business Decision Making**: Used in strategic planning and resource allocation.
- **Healthcare**: Assists in diagnosing diseases and suggesting treatment plans.
- **Finance**: Helps in credit scoring and risk assessment.
- **Marketing**: Used to segment customers and predict customer behavior.

Features	Training Data	Testing Data
Purpose	The machine-learning model is trained using training data. The more training data a model has, the more accurate predictions it can make.	Testing data is used to evaluate the model's performance.
Exposure	By using the training data, the model can gain knowledge and become more accurate in its predictions.	Until evaluation, the testing data is not exposed to the model. This guarantees that the model cannot learn the testing data by heart and produce flawless forecasts.
Use	Used iteratively to adjust model parameter.	By making predictions on the testing data and comparing them to the actual labels, the performance of the model is assessed.
Size	Typically larger	Typically smaller

Variance

- It refers to the model's sensitivity to small changes in the training data.
- A high variance means the model is too complex and learn not only the underlying pattern but also noise from training data.
- High variance models are typically overfitted.

Bias

- It refers to the error introduced by approximating a real-world problem with simplified model.
- A high bias means the model is too simple and fail to capture the underlying pattern in the data.
- High bias model are typically underfitted.

Underfitting(High Bias , Low Variance)

- It occur when a model is too simple to capture the underlying pattern of the data.
- It has high bias &low variance .
- Underfitting happens when the model has a high bias (it consistently misrepresents the data) but low variance (small change in the training data do not significantly affects the model).

Overfitting(Low Bias ,High Variance)

- It occur when a model is too complex &capture noise along with a underlying pattern in the training data.
- It occur when the model has low bias (it fits the training data well) but high variance (it is overly sensitive to small fluctuation in the training data).

Neural Networks

Neural Networks are computational models that mimic the complex functions of the human brain. The neural networks consist of interconnected nodes or neurons that process and learn from data, enabling tasks such as pattern recognition and decision making in machine learning.

Feedforward Neural Network

A Feedforward Neural Network (FNN) is a type of artificial neural network where connections between the nodes do not form cycles. It is one of the simplest forms of neural networks and is also known as a Multilayer Perceptron (MLP) when it has multiple hidden layers. In a feedforward network, the data moves in only one direction.

Key Features of a Feedforward Neural Network:

1. **Architecture**:
    - **Input Layer**:
    - **Hidden Layers**: One or more intermediate layers where the network processes inputs. Each layer consists of neurons that transform the input data by applying weights, biases, and activation functions.
    - **Output Layer**:

2. **Direction of Data Flow**:
    - In an FNN, data flows in a single direction
3. **Activation Functions**:
    - Activation functions introduce non-linearity into the network, enabling it to learn complex patterns. Common activation functions in FNNs include ReLU (Rectified Linear Unit), sigmoid, and tanh.

Problems in which ANN can be applied

ANNs are versatile tools in ML and AI that can be applied to a wide range of problems.Here are some key types of problems where ANNs can be effectively applied:

1. Classification:

- **Image Classification**: Identifying objects, people, or scenes in images .
- **Spam Detection**: Classifying emails as spam or not spam.
- **Sentiment Analysis**: Determining the sentiment (positive, negative, neutral) expressed in text data, such as reviews or social media posts.
- **Medical Diagnosis**: Classifying medical images or patient data to diagnose diseases.

2. Regression:

- **Price Prediction**: Predicting house prices, stock prices, or other financial metrics.
- **Weather Forecasting**: Predicting temperatures, precipitation, or other weather-related variables.

3. Pattern Recognition:

- **Facial Recognition**: Identifying or verifying individuals based on facial features.
- **Voice Recognition**: Identifying or verifying individuals based on their voice characteristics.
- **Handwriting Recognition**: Recognizing written characters or words from images.

4. Natural Language Processing (NLP):

- **Machine Translation**: Translating text from one language to another.
- **Chatbots and Virtual Assistants**: Understanding and generating human-like responses in conversations.

- **Text Summarization**: Creating concise summaries of longer texts.

6. Recommendation Systems:

- **Product/Content Recommendations**: Suggestingmovies, books, or music, products to customers based on their past behavior and preferences.

7. Generative Tasks:

- **Image Generation**: Creating new images based on learned patterns, such as in Generative Adversarial Networks (GANs).
- **Music and Art Creation**: Composing music or generating artworks.

8. Time Series Forecasting:

- **Financial Markets**: Predicting stock prices or economic indicators.
- **Sales Forecasting**: Predicting future sales based on historical trends.
- **Load Forecasting**: Predicting electricity or utility demand.

9. Robotics and Control Systems:

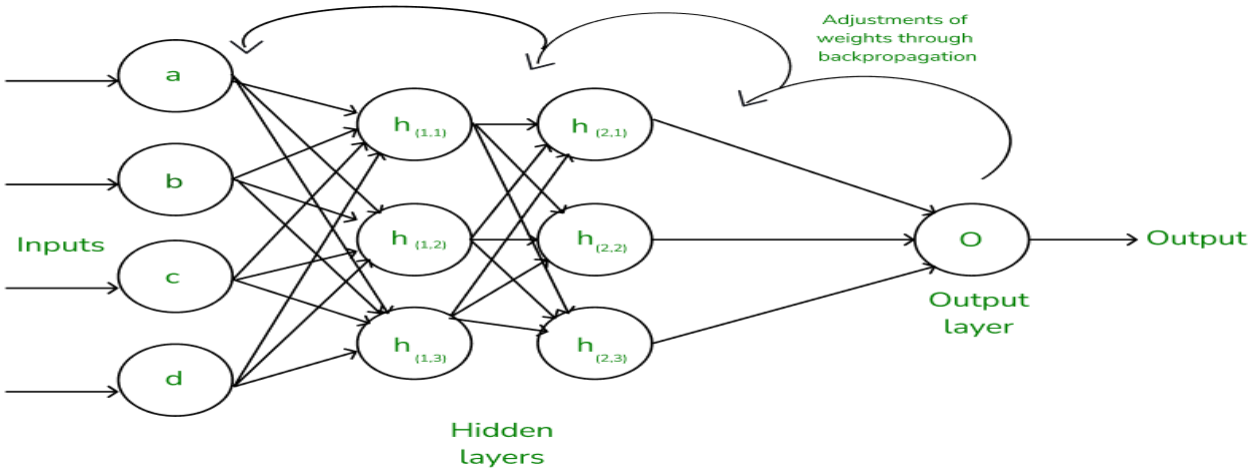
- **Autonomous Vehicles**: Navigating and making decisions in real-time.
- **Robotic Control**: Controlling robotic arms or other devices based on sensory inputs.

10. Healthcare Applications:

- **Drug Discovery**: Predicting the efficacy of new drug compounds.
- **Personalized Medicine**: Tailoring treatment plans based on individual patient data.

BackPropogation

Backpropagation is a fundamental algorithm used for training artificial NN . It involves adjusting the weights of the network based on the error of the output compared to the expected result. The goal is to minimize this error by fine-tuning the weights, thereby improving the network's performance on a given task.



Key Steps in Backpropagation:

1. **Initialization**:
   - Begin with initializing the weights and biases in the network, typically with small random values.
2. **Forward Propagation**:
   - Pass the input data through the network layer by layer, calculating the weighted sum at each neuron, applying the activation function, and obtaining the output. This produces the network's predictions.
3. **Loss Calculation**:
   - Calculate the loss (error) between the network's output and the true labels using a loss function (e.g., Mean Squared Error for regression or Cross-Entropy for classification). This loss quantifies how far the network's predictions are from the actual values.
   - **Loss = Sum (Predicted - Actual)²**
4. **Backward Propagation**:
   - **Gradient Calculation**:
     - Compute the gradient of the loss function with respect to each weight and bias in the network. This involves differentiating the loss with respect to the outputs of each neuron, then propagating this gradient backwards through the network using the chain rule.
   - **Weight Update**:
     - Adjust the weights and biases using the computed gradients. The adjustment is typically done using an optimization algorithm such as Gradient Descent, with the weights being updated as follows:
     - $w_i := w_i - \eta \partial L / \partial w$
     - $b := b - \eta \partial L / \partial b$

        - where wi are the weights, b are the biases, L is the loss function, and η is the learning rate. The learning rate controls the step size of the update.
2. **Iteration and Convergence**:
    - Repeat the forward and backward propagation steps for multiple epochs (iterations over the entire training dataset) until the network's performance reaches a satisfactory level or the loss stabilizes, indicating convergence.

Concepts and Terminology:

- **Gradient Descent**: An optimization algorithm used to minimize the loss function by iteratively moving towards the steepest descent as defined by the negative gradient.
- **Learning Rate (η\etaη)**: A hyperparameter that controls how much the model's weights are adjusted with respect to the loss gradient. A small learning rate may result in slow convergence, while a large learning rate may cause the model to converge too quickly to a suboptimal solution or even diverge.
- **Overfitting**: A scenario where the network learns the training data too well, including the noise, resulting in poor generalization to new data. Techniques like regularization, dropout, and cross-validation are used to mitigate overfitting.

Limitations:

While backpropagation is powerful, it has some limitations:

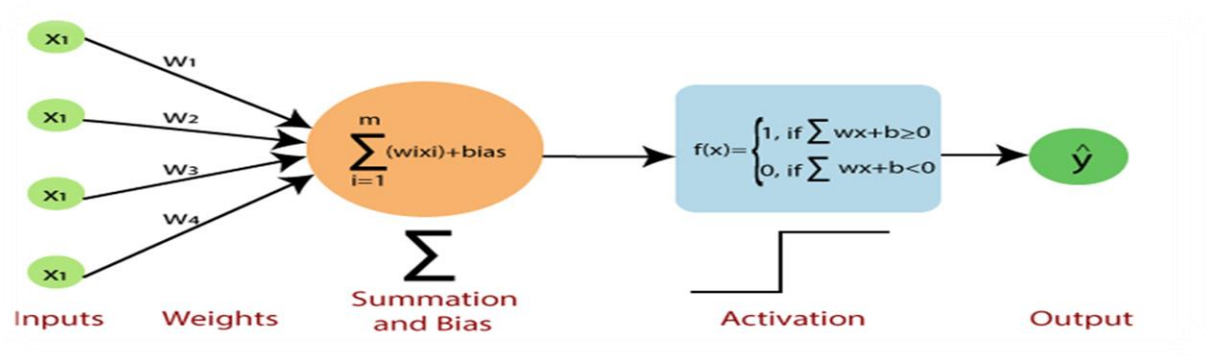
- It can be computationally expensive, especially with deep networks.
- It requires careful tuning of hyperparameters like the learning rate.

Perceptron

The Perceptron is a type of artificial neural network that serves as a basic building block for more complex neural networks. It is one of the simplest forms of neural networks and is used primarily for binary classification tasks.

Structure of a Perceptron

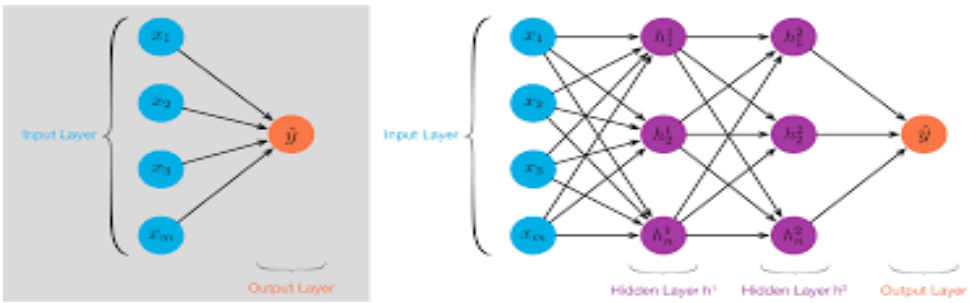
A perceptron consists of a single layer of neurons, known as the output layer, and an input layer. There are no hidden layers in the simplest form of a perceptron.



Key Components:

1. **Input Features (xi)**:
   - The perceptron receives multiple input features, represented as a vector [x1,x2,…,xn,where each xi corresponds to an attribute of the input data.
2. **Weights (wi)**:
   - Each input feature is associated with a weight wi. These weights determine the influence of each input feature on the output.
3. **Bias (b)**:
   - A bias term b is included to adjust the output independently of the input values. It helps in shifting the decision boundary to better fit the data.
4. **Weighted Sum (z)**:
   - The perceptron calculates a weighted sum of the input features, including the bias:
   - z=wixi+b.
5. **Activation Function**:
   - The perceptron uses a step activation function to produce a binary output. The activation function is applied to the weighted sum z to determine the output y:
   - y=sign(z)={1if z≥0 and 0 if z<0

Single Layer Perceptron vs Multilaper Perceptron



Feature	Single-Layer Perceptron (SLP)	Multi-Layer Perceptron (MLP)
Structure	Input layer and output layer only	Input layer, one or more hidden layers, and output layer
Neurons	Neurons in the output layer only	Neurons in input, hidden, and output layers
Capabilities	Only handles linearly separable data	Handles both linear and non-linear data
Learning Rule	Perceptron Learning Rule	Backpropagation algorithm
Activation Function	Step function	Non-linear functions (ReLU, sigmoid, tanh, etc.)
Complexity	Simple, fast training	More complex, longer training time
Non-Linear Capability	Not capable	Capable of modeling complex, non-linear relationships
Training Requirements	Less computational power and time	Requires more computational power and time
Risk of Overfitting	Lower	Higher, especially with many parameters; requires regularization
Applications	Simple binary classification, linearly separable data	Complex classification, regression, and tasks like image/speech recognition
Limitations	Cannot model non-linear boundaries	Prone to overfitting, vanishing/exploding gradients in deep networks

Issues in machine learning

1. **Data Quality and Quantity**
   - **Insufficient Data**: Not enough training data.
   - **Poor Quality Data**: Noisy or inaccurate data.
2. **Overfitting and Underfitting**
   - **Overfitting**: Model learns too much detail, including noise.
   - **Underfitting**: Model is too simple to capture patterns.
3. **Feature Engineering**
   - **Feature Selection**: Choosing relevant features.
   - **Feature Extraction**: Creating new informative features.
4. **Model Complexity and Interpretability**
   - **Complexity**: Complex models are harder to interpret.
   - **Interpretability**: Understanding model decisions.

5. **Computational Cost**
    - **Training Time**: Long training periods.
    - **Resource Requirements**: High computational needs.
6. **Bias and Fairness**
    - **Algorithmic Bias**: Inherited biases from data.
    - **Fairness**: Avoiding discriminatory outcomes.

Role of Bias in ANN

1. **Shifting Activation Function**: Bias adjusts the neuron's activation function, allowing the network to learn patterns not centered around zero.
2. **Control over Output**: It provides control over neuron activation, enabling the neuron to activate even if inputs are zero.
3. **Learning Ability**: Biases contribute to learning non-linear patterns and complex relationships in data.
4. **Training Impact**: Bias terms are learned alongside weights, enhancing the network's flexibility and accuracy.

Gradient Descent Delta Rule

The Gradient Descent Delta Rule, often simply called the Delta Rule, is a fundamental learning rule used in training artificial neural networks. It adjusts the weights of the network to minimize the error in predictions. Here's a brief explanation:

Gradient Descent Delta Rule

Purpose:

The Delta Rule aims to minimize the difference between the predicted output of a neural network and the actual target value by adjusting the network's weights.

Process:

1. **Initialize Weights**: Start with small random weights.
2. **Compute Output**: For each input, calculate the output using the current weights.
3. **Calculate Error**: Determine the error by subtracting the predicted output from the actual target value:

   $$\text{Error} = \text{Target} - \text{Output}$$

4. **Update Weights**: Adjust the weights in the direction that reduces the error. The weight update rule is given by:

   $$\Delta w = \eta \cdot \text{Error} \cdot x$$

Where:

- Δw is the change in weight.
- η is the learning rate, a small positive constant.
- Error is the difference between the target and the output.
- x is the input feature associated with the weight.

5. **Iterate**: Repeat the process for many iterations (epochs) until the error is minimized or converges to an acceptable level.

Algorithmic Convergence in ANN

- Algorithmic convergence in the context of ANN refers to the process by which the training algo adjust the network parameters (weight & bias) to minimize a defined objective function.
- The goal is to find the set of parameter that best fit the training data.

Generalization Property

- It refers to the ability of a trained model to perform well on unseen data, beyond the training example it was trained on.
- It is a critical measure of how well the model has learned from the training data and how well it can adapt to new unseen input.