**Course 2: Intelligent Systems**

**Decision Trees:**

**Aim:** Decision trees aim to classify or predict outcomes based on a set of attributes.

**Definition:** A decision tree is a tree-like model where each internal node represents an attribute, each branch represents a decision rule, and each leaf node represents the outcome or class label. It can be used for both classification and regression problems.

**Example:** Suppose we want to predict whether a loan application will be approved or not based on attributes like ID, age, owns\_house, and credit\_rating. A decision tree can help make decisions by evaluating these attributes and their relationships with the approval outcome.



**Information Gain:** Information gain is a measure used in decision tree algorithms to select the best attribute at each step. It quantifies the amount of information gained by partitioning the data based on a particular attribute.

**Pruning:** Pruning is a technique used to reduce the size of a decision tree by removing branches or nodes that do not contribute significantly to the accuracy of the model. It helps prevent overfitting and improves generalization.

**Artificial Neural Networks:**

**Aim:** Artificial Neural Networks (ANNs) aim to simulate the behavior of biological neurons to solve complex problems.

**Definition:** ANNs consist of interconnected nodes (neurons) organized in layers. Each neuron receives inputs, applies an activation function to produce an output, and passes it to the next layer. ANNs are capable of learning from data and adjusting the strengths of connections (weights) between neurons.

**Example:**Imagine a neural network that recognizes handwritten digits. Each neuron in the network represents a pixel of an image, and the network learns patterns and features to classify the digit based on these pixel values.

**Perceptron Algorithm (Pseudocode for a Single Neuron Perceptron):**



**Testing Algorithm:**



**Neural Network Processing:**

* Nodes: Nodes in a neural network are also known as neurons. Each node takes inputs, applies an activation function to produce an output, and passes it to the next layer of neurons.
* Layers: Neural networks are organized in layers. The input layer receives the initial inputs, hidden layers process the inputs, and the output layer produces the final outputs.

**Activation Functions:** Activation functions determine the output of a neuron based on its weighted inputs. Some common activation functions include:

* Constant Activation: The output is a constant value.
* Step Activation: The output is binary (0 or 1) based on a threshold.
* Linear Activation: The output is a linear function of the inputs.
* Sigmoid Activation: The output is a smoothed binary value between 0 and 1.
* Gaussian Activation: The output follows a Gaussian (bell-shaped) distribution.

**Gradient Descent:** Gradient descent is an optimization algorithm used to adjust the weights in a neural network to minimize the error between predicted and actual outputs. It works by iteratively updating the weights in the direction of the steepest descent of the error surface.

**Computing the Error's Gradient:** The error gradient represents the rate of change of the error with respect to the weights. It is used to update the weights during the training process.

**Neuron Learning:** Neuron learning refers to the process of adjusting the weights and biases of a neuron to improve its performance. It involves iterative training using input-output pairs and updating the weights based on the error between predicted and actual outputs.

**Simple Gradient Descent (SGD):**

* Simple Gradient Descent is an optimization algorithm used to update the weights in a neural network based on the gradient of the error function with respect to the weights.
* It works by calculating the gradient of the error function for the entire training dataset and updating the weights in the opposite direction of the gradient.
* The weight update rule in SGD can be represented as:
* new\_weight = old\_weight - learning\_rate \* gradient
* where learning\_rate controls the size of the weight update and gradient represents the gradient of the error function with respect to the weight.

**Stochastic Gradient Descent (SGD):**

* Stochastic Gradient Descent is a variation of the SGD algorithm where the weight updates are performed on individual training samples rather than the entire dataset.
* It randomly selects a single training sample, calculates the gradient of the error function for that sample, and updates the weights based on the gradient.
* The weight update rule in SGD can be represented as:
* new\_weight = old\_weight - learning\_rate \* gradient\_i
* where learning\_rate controls the size of the weight update and gradient represents the gradient of the error function with respect to the weight for a single training sample.
* SGD is computationally more efficient than SGD because it avoids calculating the gradients for the entire dataset in each iteration.
* However, it introduces more stochasticity due to the random selection of training samples, which can sometimes lead to noisy weight updates.

In both SGD and SGD, the learning rate determines the step size of weight updates. It is a hyperparameter that needs to be carefully tuned to ensure the convergence and stability of the learning process.