U2

Part-1: Artificial Neural Networks-1:

Intro:

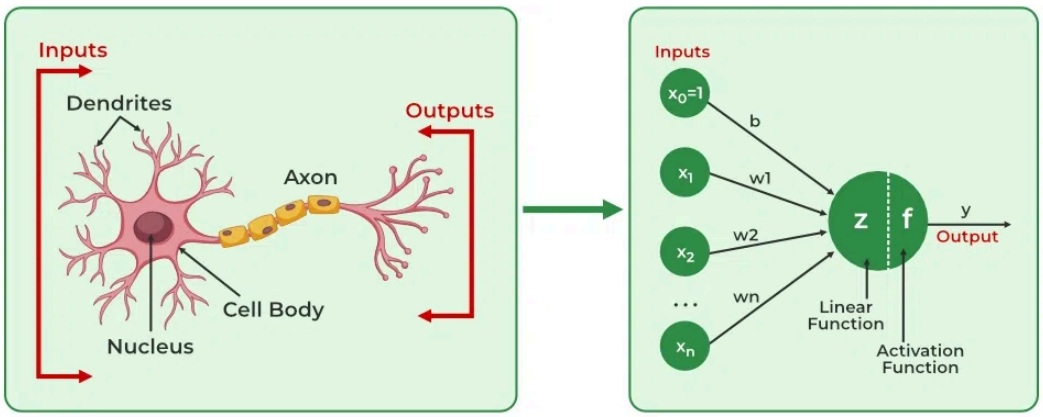
Artificial Neural Networks (ANNs)

1. Introduction

- **Definition:** ANNs are computational models inspired by the human brain's network of neurons. They are used for tasks such as pattern recognition, classification, and prediction.
- **Components:**
  - **Neurons:** Basic units of computation.
  - **Layers:** Include input layers, hidden layers, and output layers.
  - **Weights:** Parameters that are adjusted during training to minimize errors.

2. Structure of ANNs

Biological Neuron	Artificial Neuron
Dendrite	Inputs
Cell nucleus or Soma	Nodes
Synapses	Weights
Axon	Output



- **Neurons:**
  - **Input Neurons:** Receive external data.
  - **Hidden Neurons:** Perform computations and feature extraction.
  - **Output Neurons:** Produce final results.
- **Layers:**
  - **Input Layer:** Receives input features.
  - **Hidden Layers:** Intermediate layers where computation occurs. Multiple hidden layers make the network deep.
  - **Output Layer:** Provides the final prediction or classification.

3. Activation Functions

- **Purpose:** Introduce non-linearity into the network, allowing it to learn complex patterns.
- **Common Functions:**

- **Sigmoid:** $f(x) = \frac{1}{1+e^{-x}}$
- **ReLU (Rectified Linear Unit):** $f(x) = \max(0, x)$
- **Tanh:** $f(x) = \tanh(x)$
- **Softmax:** Converts outputs into probability distributions.

4. Training Process

- **Forward Propagation:** Input data is passed through the network, layer by layer, to produce an output.
- **Loss Function:** Measures the difference between predicted and actual values.
  - **Common Loss Functions:** Mean Squared Error (MSE), Cross-Entropy Loss.
- **Backpropagation:** Algorithm for adjusting weights based on error gradients to minimize the loss function.
  - **Gradient Descent:** Optimization method used to update weights.

5. Optimization Algorithms

- **Stochastic Gradient Descent (SGD):** Updates weights using random subsets of data.
- **Adam:** Adaptive optimization algorithm combining advantages of other methods (e.g., momentum, adaptive learning rates).
- **RMSprop:** Adjusts the learning rate based on recent gradients.

6. Applications

- **Image and Speech Recognition:** Identify objects and transcribe spoken language.
- **Natural Language Processing:** Tasks like translation, sentiment analysis, and text generation.
- **Predictive Analytics:** Forecasting trends and patterns in various domains.

7. Challenges

- **Computational Resources:** Training deep networks requires substantial computational power.

- **Data Requirements:** Large datasets are often needed for effective training.
- **Overfitting:** Complex models may perform well on training data but poorly on unseen data.

Neural Network Representation:

1. What are Neural Networks?

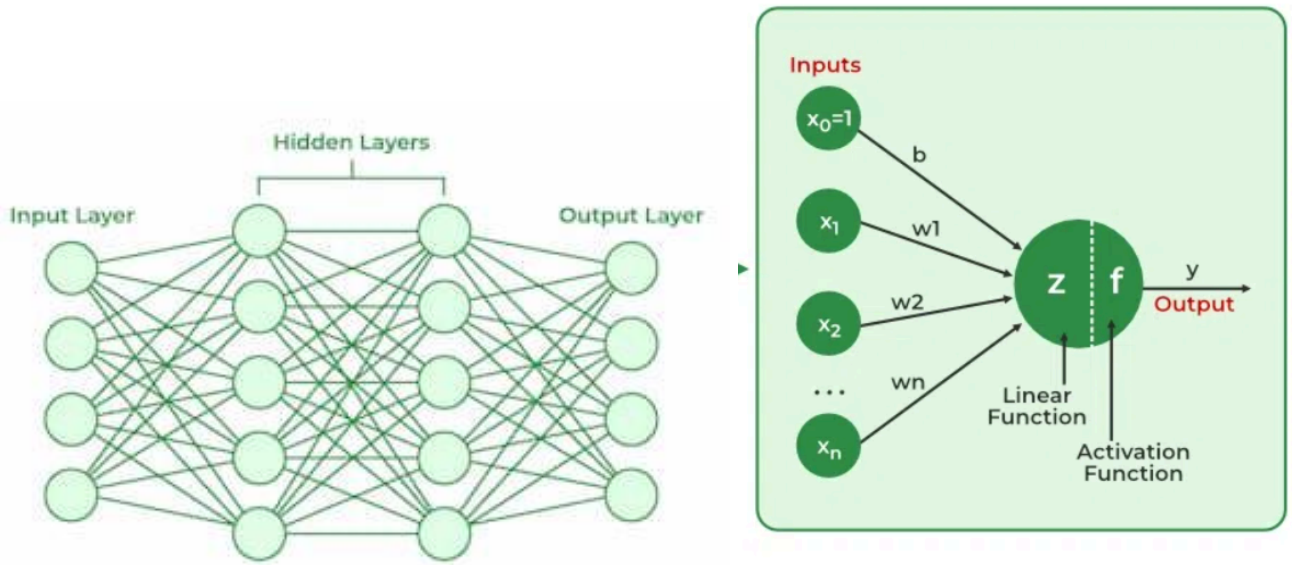
Neural networks are computational models inspired by the structure and function of the human brain. They are designed to recognize patterns, classify data, and make predictions based on input data. Neural networks consist of interconnected nodes or "neurons," organized into layers, and are used in various applications including image and speech recognition, natural language processing, and predictive analytics.

2. Importance of Neural Networks

- **Pattern Recognition:** Neural networks excel at identifying patterns and relationships in complex data, making them suitable for tasks like image and speech recognition.
- **Adaptability:** They can learn from data and adapt to new, unseen patterns, improving their performance over time with more data.
- **Feature Extraction:** Automatically extract and learn relevant features from raw data without needing manual feature engineering.
- **Versatility:** Applicable to a wide range of problems, from classification and regression to time series forecasting and anomaly detection.
- **Deep Learning:** Enable the development of deep learning models with multiple layers, capable of handling large-scale data and complex tasks.

3. How Neural Networks Work

Neural networks operate through two main processes: forward propagation and backpropagation.



Forward Propagation

1. **Input Data:** Data is fed into the input layer, which passes it to the next layer.

2. **Weighted Sum:** Each neuron in a layer computes a weighted sum of its inputs. The weighted sum is calculated as:

$$z_i = \sum_j w_{ij} x_j + b_i$$

where **wij** is the weight from neuron **j** in the previous layer to neuron **i** in the current layer, **xj** is the input, and bi is the bias term.

3. **Activation Function:** The weighted sum is passed through an activation function to introduce non-linearity. The activation function **f(zi)** determines the neuron's output:

$$a_i = f(z_i)$$

Common activation functions include:

- Sigmoid: $f(z) = \frac{1}{1+e^{-z}}$
- ReLU (Rectified Linear Unit): $f(z) = \max(0, z)$
- Tanh: $f(z) = \tanh(z)$

4. **Propagation Through Layers:** This process continues through all layers of the network until the final output layer is reached, producing the network's prediction.

Backpropagation

1. **Compute Loss:** The network's output is compared to the actual target value using a loss function (e.g., Mean Squared Error for regression or Cross-Entropy Loss for classification). The loss function measures the discrepancy between the predicted and actual values.
2. **Calculate Gradients:** Using the loss function, the gradient of the loss with respect to each weight is calculated. This involves applying the chain rule of calculus to compute how changes in weights affect the loss.
3. **Update Weights:** The weights are updated using an optimization algorithm such as Gradient Descent. The update rule for a weight **wij** is:

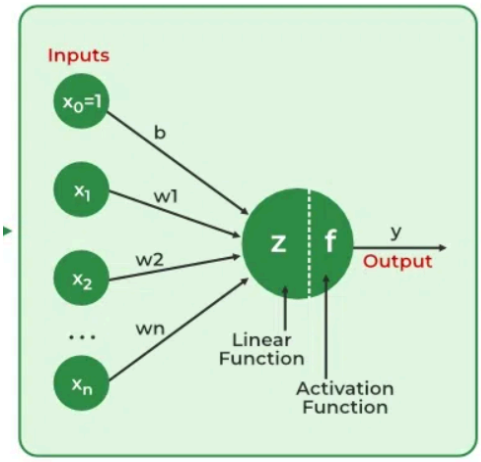
$$w_{ij} = w_{ij} - \eta \frac{\partial L}{\partial w_{ij}}$$

where η is the learning rate, and **∂L\∂wij** is the gradient of the loss function with respect to the weight.

4. **Iterate:** This process of forward propagation and backpropagation is repeated iteratively through multiple epochs (complete passes through the training dataset) until the network's performance improves and converges to a satisfactory level.

Perceptions:

- A human brain is made of billions of neurons which are the basic building blocks of the neural network.
- The decision making in the human brain is possible because of the neural network.
- Each neuron takes some form of input through an electrical signal and produces an output, which in turn is the input to another neuron and so on.
- In the ANN, the artificial neuron (a unit of ANN) is termed a **perceptron.**



- The idea of a perceptron is to use different weights to represent the importance of each input, and that the sum of the values should be greater than a threshold value before making a decision like yes or no (true or false) (0 or 1).

Structure of a Perceptron

A perceptron consists of:

- **Inputs:** Features of the data. (weights and the bias)
- **Weights:** Each input has an associated weight that determines the importance of that input.
- **Bias:** A constant added to the weighted sum of inputs to adjust the decision boundary. This is the value which is altered in the back propagation process to train the ANN model, iteratively. Initially it is set to (0) or (1), and changed iteratively based on the difference b/w expected and actual output.
- **Linear Function:** A mathematical function that produces a straight-line graph and is used to compute weighted sums of inputs before applying an activation function. It is given by:

$$z = \sum_i w_i x_i + b$$

Where:

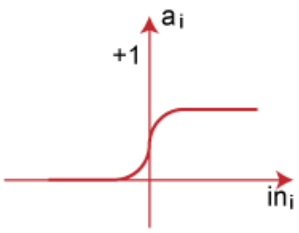
- **wi** are the weights.
- **xi** are the input values.
- **b** is the bias term.
- **z** is the weighted sum (also known as the pre-activation value).

- **Activation Function:** A function applied to the weighted sum to produce the output. It introduces non-linearity into the neural network, allowing it to model complex relationships and patterns that cannot be captured by linear functions alone. After computing the weighted sum (linear function), the activation function transforms this value to produce the output of the neuron.

  Common Activation Functions are:

1. **Sigmoid Function:**
    - **Formula:**

$$f(z) = \frac{1}{1+e^{-z}}$$



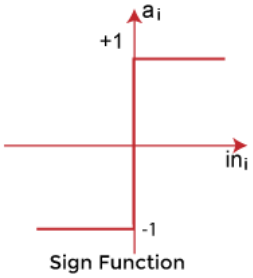
Sigmoid Function

    - **Range:** (0, 1)
    - **Characteristics:** Maps any real-valued number to a value between 0 and 1, making it useful for binary classification problems.
    - **Pros:** Smooth gradient, which helps in training.
    - **Cons:** Can cause vanishing gradient issues during training (gradients become very small).
2. **ReLU (Rectified Linear Unit):**
    - **Formula:**

$$f(z) = \max(0, z)$$

    - **Range:** [0, ∞)
    - **Characteristics:** Outputs zero for negative inputs and passes positive inputs unchanged. It's widely used in hidden layers of deep networks.
    - **Pros:** Reduces the likelihood of vanishing gradients, computationally efficient.
    - **Cons:** Can lead to dead neurons (neurons that always output zero) if not managed properly.
3. **Tanh (Hyperbolic Tangent):**
    - **Formula:**

$$f(z) = \tanh(z)$$



Sign Function

    - **Range:** (-1, 1)
    - **Characteristics:** Maps input values to a range between -1 and 1, centering the data and helping with training stability.
    - **Pros:** Centered around zero, which can make optimization easier.
    - **Cons:** Can suffer from vanishing gradient problems.

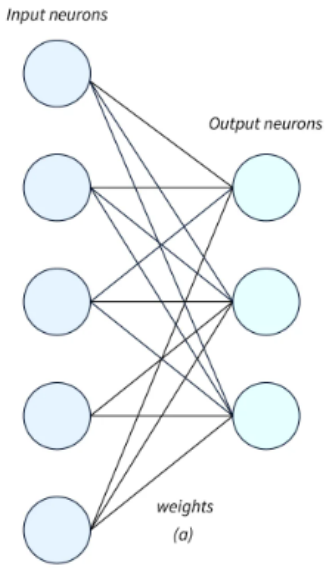
- So, the output of a single perceptron is nothing but the output of the activation function, which takes the weighted-sum(z) and the input.

Types of Perceptrons:

Perceptrons come in various forms and configurations, each suited for different types of tasks and complexities.

1. Single-Layer Perceptron

Description:



- The single-layer perceptron consists of one layer of neurons, also known as the input layer. It directly connects inputs to outputs with no hidden layers.
- It is capable of **solving linearly separable problems**, where classes can be separated by a straight line or hyperplane.

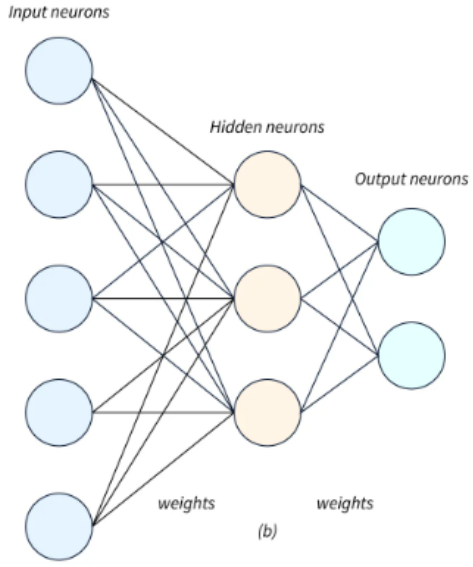
Example Use Case:

- Binary classification problems, such as distinguishing between two classes of data points that are linearly separable
  (e.g., classifying points as above or below a line).

Limitations:

- Cannot solve problems that are not linearly separable (e.g., the XOR problem).

2. Multi-Layer Perceptron (MLP)



Description:

- MLPs, also known as feedforward neural networks, consist of one or more hidden layers between the input and output layers.
- Each layer in an MLP is fully connected to the next layer, meaning each neuron in one layer is connected to every neuron in the next layer.
- It is used in **solving the non-linearly separable problems.**

Example Use Case:

- Complex classification and regression problems where data is not linearly separable. MLPs can model non-linear relationships due to the multiple layers and non-linear activation functions.

Architecture:

- **Input Layer:** Receives the input features.
- **Hidden Layers:** One or more layers where neurons perform non-linear transformations.
- **Output Layer:** Produces the final predictions or classifications.

<u>Multi-Layer Networks:</u>

- Multi-layer networks, often referred to as Multi-Layer Perceptrons (MLPs), are a type of artificial neural network that consists of multiple layers of neurons(input, hidden and output).

- These networks are capable of learning complex patterns and making predictions based on the input data.

- There are two methods to train the multilayer network of perceptrons:

(a) Forward Propagation
(b) Backward Propagation

a. Forward Propagation (Propagating Inputs from input layer to output layer, through hidden layer)

Purpose: Forward propagation is the process by which input data is passed through the network to generate predictions or outputs. It involves calculating the activations of neurons layer by layer from the input layer through the hidden layers to the output layer.

Steps:

1. **Input Layer:**
   - The network receives the input features x=[x1,x2,...,xn].
2. **Hidden Layers:**
   - For each neuron jjj in a hidden layer, compute the weighted sum of inputs and add the bias:

$$z_j = \sum_i w_{ji} x_i + b_j$$

   - Apply the activation function fff to the weighted sum to get the activation **aj:**

$$a_j = f(z_j)$$

3. **Output Layer:**
   - Compute the weighted sum for the output layer neurons and apply the activation function (e.g., softmax for classification):

$$z_k = \sum_j w_{kj} a_j + b_k$$

$$a_k = f(z_k)$$

   - The final output **ak** represents the network's prediction.
   - This output is compared to the target to compute the loss.

2. Backward Propagation (Propagating error from output layer to input layer through hidden layer and updating the biases accordingly)

Purpose: Backward propagation (backpropagation) is the process used to update the network's weights and biases based on the error calculated from the forward propagation. It involves calculating the gradients of the loss function with respect to each weight and bias and using these gradients to adjust the weights and biases to minimize the loss.

Steps:

1. **Compute Loss(Error):**
   - Calculate the loss (error) between the predicted output and the actual target using a loss function.
2. Compute the updated values of all weights and biases, from:
   a. Output Layer —---------> Hidden Layer

b. Hidden Layer ——----------> Input Layer

Remarks on the Backpropagation Algorithm

The backpropagation algorithm is a cornerstone of training neural networks. Here are some key remarks and considerations regarding its implementation and performance:

1. Gradient Calculation

- **Chain Rule:** Backpropagation relies on the chain rule of calculus to compute gradients efficiently. This involves calculating the gradient of the loss function with respect to each weight by propagating errors backward through the network.
- **Partial Derivatives:** It computes partial derivatives of the loss function with respect to each weight and bias, allowing the network to adjust these parameters to minimize the error.

2. Learning Rate

- **Choosing Learning Rate:** The learning rate (η\etaη) determines the size of the steps taken during gradient descent. A learning rate that is too high can lead to overshooting, while a rate that is too low can result in slow convergence.
- **Adaptive Learning Rates:** Techniques like Adam, RMSprop, and AdaGrad use adaptive learning rates to adjust the step size based on the gradients, improving convergence.

3. Convergence

- **Local Minima:** Gradient descent methods used in backpropagation can get stuck in local minima. Advanced techniques like stochastic gradient descent (SGD) with momentum or global optimization methods can help escape local minima.
- **Convergence Criteria:** Training continues until the loss function converges (i.e., changes become very small) or a predefined number of epochs is reached.

4. Overfitting

- **Regularization:** Regularization techniques like dropout, L2 regularization, and early stopping help prevent overfitting by discouraging the model from becoming too complex and fitting noise in the training data.
- **Cross-Validation:** Using cross-validation techniques helps to ensure that the model generalizes well to unseen data.

5. Computational Efficiency

- **Batch Size:** The choice of batch size (the number of training examples used to compute the gradient) can impact training efficiency. Smaller batches can lead to noisy updates but may help in generalization, while larger batches lead to more stable but computationally intensive updates.
- **Parallelization:** Backpropagation computations can be parallelized across multiple CPUs or GPUs, significantly speeding up training.

6. Activation Functions

- **Non-Linearity:** The choice of activation function affects the network's ability to learn complex patterns. Functions like ReLU, sigmoid, and tanh each have different characteristics that can impact training dynamics.
- **Vanishing/Exploding Gradients:** Some activation functions, like sigmoid and tanh, can lead to vanishing gradients, where gradients become very small, slowing down training. ReLU and its variants can help mitigate this issue.

7. Initialization of Weights

- **Importance of Initialization:** Proper weight initialization is crucial for effective training. Poor initialization can lead to slow convergence or difficulty in training. Techniques like He initialization for ReLU activations or Xavier initialization for tanh activations can improve convergence.
- **Random Initialization:** Weights are typically initialized with small random values to break symmetry and ensure that neurons learn different features.

Evaluating Hypothesis:

- Evaluation of a Hypothesis involves assessing whether the data and evidence support(proven to be true) or refute(proven to be false) a proposed hypothesis.

- Evaluation steps for Hypothesis:

1. Formulating Hypotheses

Hypotheses: A hypothesis is a testable statement or prediction about the relationship between variables. It can be:

- **Null Hypothesis (H_0):** The default assumption that there is no effect or no difference.
- **Alternative Hypothesis (H_1 or Ha):** The statement that there is an effect or a difference.

Example:

- **Null Hypothesis (H_0):** There is no difference in test scores between students who study with music and those who study without music.
- **Alternative Hypothesis (H_1):** Students who study with music have different test scores compared to those who study without music.

2. Designing the Study

Data Collection: Collect data that will help test the hypothesis. Ensure that the data is relevant, reliable, and sufficient to draw valid conclusions.

3. Analyzing Data

Statistical Tests: Use appropriate statistical tests to analyze the data. The choice of test depends on the type of data and the hypothesis. Common tests include:

- **t-Test:** Compares the means of two groups.
- **Chi-Square Test:** Tests the association between categorical variables.

4. Interpreting Results

Compare p-Value to Level of Significance(α):
[here p-value is the probability (test result) that the null hypothesis is true]

- <u>Reject Null Hypothesis</u>: If the p-value is less than α (e.g., 0.05), there is sufficient evidence to reject the null hypothesis in favor of the alternative hypothesis.

- <u>Fail to Reject Null Hypothesis</u>: If the p-value is greater than α, there is insufficient evidence to reject the null hypothesis.

<u>Basics of Sampling Theory:</u>

1. Random Variable

Definition: A random variable is a variable whose value is subject to variations due to chance. It can take on different values with certain probabilities.

- **Discrete Random Variable**: Takes on a countable number of distinct values (e.g., the number of heads in a series of coin flips).
- **Continuous Random Variable**: Takes on an uncountable number of values within a range (e.g., height, weight).

2. Probability Distribution

Definition: A probability distribution describes how probabilities are distributed over the possible values of a random variable. It shows the likelihood of each outcome.

- **Discrete Probability Distribution**: Includes distributions like the Binomial distribution or Poisson distribution.
- **Continuous Probability Distribution**: Includes distributions like the Normal distribution or Exponential distribution.

3. Mean (or Expectation)

Definition: The mean (or expectation) of a random variable is the average value it would take over many trials.

- Discrete Random Variable: $\mathbb{E}[X] = \sum_i x_i \cdot P(x_i)$
- Continuous Random Variable: $\mathbb{E}[X] = \int_{-\infty}^{\infty} x \cdot f(x)\, dx$

where $f(x)$ is the probability density function.

4. Variance

Definition: The variance measures how much the values of a random variable spread out from the mean. It is the expected value of the squared deviation from the mean.

- **Formula:** $\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$

5. Standard Deviation

Definition: The standard deviation is the square root of the variance. It provides a measure of the dispersion of a set of values.

- **Formula:** $\sigma = \sqrt{\text{Var}(X)}$

6. Binomial Distribution

Definition: The Binomial distribution models the number of successes in a fixed number of independent Bernoulli trials, each with the same probability of success.

- **Probability Mass Function (PMF):** $P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$
  - n = number of trials
  - k = number of successes
  - p = probability of success in each trial

7. Normal Distribution

Definition: The Normal distribution, or Gaussian distribution, is a continuous probability distribution characterized by its bell-shaped curve. It is defined by its mean (μ) and standard deviation (σ).

- **Probability Density Function (PDF):** $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

8. Central Limit Theorem (CLT)

Definition: The CLT states that the sampling distribution of the sample mean approaches a normal distribution as the sample size becomes large, regardless of the original distribution of the data.

- **Implication**: For a sufficiently large sample size, the distribution of the sample mean will be approximately normal, even if the population distribution is not.

9. Estimator

Definition: An estimator is a rule or method for estimating an unknown parameter based on sample data. For example, the sample mean is an estimator of the population mean.

10. Estimation Bias

Definition: Estimation bias refers to the difference between the expected value of an estimator and the true value of the parameter being estimated.

- **Bias:** $\text{Bias}(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta$
  - $\hat{\theta}$ = estimator
  - θ = true parameter value

11. Confidence Interval

Definition: A confidence interval provides a range of values within which the true parameter value is expected to lie, with a certain level of confidence.

- **Formula for Confidence Interval:** For a mean with a normal distribution, the 95% confidence interval is typically $\bar{X} \pm z \cdot \frac{\sigma}{\sqrt{n}}$
  - $\bar{X}$ = sample mean
  - z = z-score corresponding to the desired confidence level
  - σ = standard deviation
  - n = sample size

General approach for deriving confidence intervals:

1. Define the Parameter and Hypothesis

- **Identify the Parameter**: Determine what parameter you want to estimate (e.g., population mean, proportion).
- **State the Hypothesis**: Formulate the null and alternative hypotheses if you're performing hypothesis testing.

2. Choose the Confidence Level

- **Confidence Level**: Select the confidence level for the interval, commonly 95% or 99%. This defines the probability that the interval will contain the true parameter value.

3. Collect and Prepare Data

- **Sample Data**: Gather and prepare your sample data. Ensure that it is representative of the population and meets necessary assumptions (e.g., normality for certain methods).

4. Determine the Distribution and Statistic

- **Sampling Distribution**: Identify the sampling distribution of the estimator. Common distributions include:
    - **Normal Distribution**: For large samples or when the population variance is known.
    - **t-Distribution**: For small samples where the population variance is unknown.
- **Sample Statistic**: Calculate the sample statistic you are using to estimate the parameter (e.g., sample mean, sample proportion).

5. Find the Critical Value

- **Critical Value (z or t)**: Determine the critical value associated with the chosen confidence level.
    - **For Normal Distribution**: Use z-scores (e.g., z≈1.96z \approx 1.96z≈1.96 for 95% confidence).
    - **For t-Distribution**: Use t-scores, which depend on the sample size and degrees of freedom.

6.Construct the Confidence Interval

- **Confidence Interval**: Add and subtract the margin of error from the sample statistic to get the interval.

- **Formula for Confidence Interval:** For a mean with a normal distribution, the 95% confidence interval is typically $\bar{X} \pm z \cdot \frac{\sigma}{\sqrt{n}}$
    - $\bar{X}$ = sample mean
    - z = z-score corresponding to the desired confidence level
    - σ = standard deviation
    - n = sample size

Difference in Error of two Hypotheses:

When comparing two hypotheses, the difference in error can be understood in terms of how well each hypothesis performs in relation to the true underlying distribution or data. This comparison often involves assessing various types of errors, depending on the context. Here's how you can approach it:

1. Define the Hypotheses

- **Null Hypothesis (H0H_0H0)**: A statement or default position that there is no effect or no difference.
- **Alternative Hypothesis (H1H_1H1 or HAH_AHA)**: A statement that contradicts the null hypothesis, suggesting that there is an effect or a difference.

2. Identify the Types of Errors

In hypothesis testing, errors are categorized as:

- **Type I Error (False Positive)**: Rejecting the null hypothesis when it is actually true. Denoted by α\alphaα, this is the significance level of the test.

- **Type II Error (False Negative)**: Failing to reject the null hypothesis when the alternative hypothesis is true. Denoted by β\betaβ.

3. Compare Errors for Two Hypotheses

To compare the errors of two hypotheses, consider the following aspects:

- **Error Rates**: Calculate the Type I and Type II errors for each hypothesis if they are testable. The error rates can be assessed using:
    - **Power of the Test**: The probability of correctly rejecting a false null hypothesis (1 - β\betaβ).
    - **Significance Level**: The probability of incorrectly rejecting a true null hypothesis (α\alphaα).
- **Performance Metrics**: For models or estimators, evaluate metrics such as:
    - **Bias**: The difference between the expected value of the estimator and the true parameter value.
    - **Variance**: The variability of the estimator across different samples.

4. Conduct Statistical Tests

To determine if there is a significant difference between the errors of two hypotheses, you can:

- **Perform Hypothesis Tests**: Use statistical tests to compare the performance of two hypotheses or models. For example:
    - **t-Test**: To compare the means of two groups.
    - **Chi-Square Test**: To compare categorical distributions.

5. Interpret Results

- Determine if the difference in errors is statistically significant. This involves looking at p-values and confidence intervals for the differences in performance metrics.