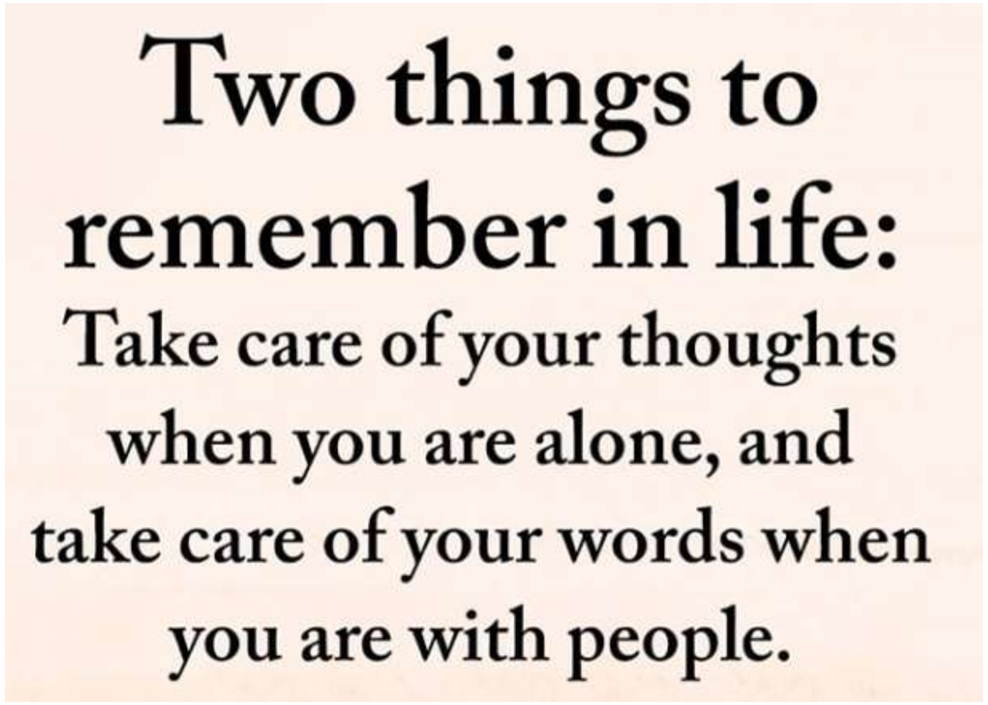
OTDM UNIT IV

Dr Ravi Prakash Shahi

8979048096, ravishahi71@gmail.com


Two things to remember in life:

Take care of your thoughts when you are alone, and take care of your words when you are with people.

Newton's Method in Optimization

Newton's Method (or the Newton–Raphson method) is a **second-order iterative optimization technique** used to find **stationary points** (minima, maxima, or saddle points) of a real-valued differentiable function $f(x)$.

It extends the 1D Newton–Raphson root-finding method to optimization problems by finding where the **gradient** (first derivative) becomes zero.

Objective

We want to find x^* such that:

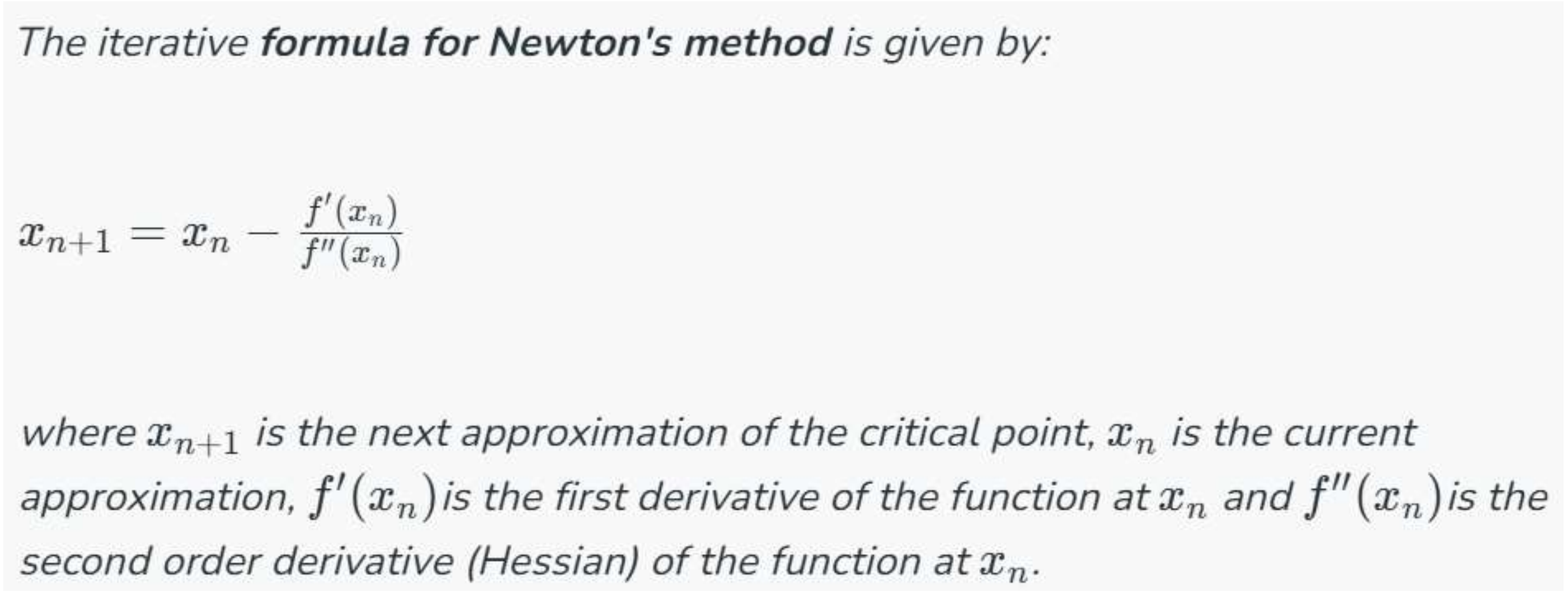
$$\nabla f(x^*) = 0$$

where

- $\nabla f(x)$ = gradient vector of $f(x)$,
- $\nabla^2 f(x)$ = Hessian matrix (matrix of second derivatives).

Newton's Method in Optimization

We are interested in finding critical points of the function where the first derivative is zero (for minima or maxima). Newton's method utilizes the first and second derivatives of the function to iteratively refine the solution.



The iterative ***formula for Newton's method*** *is given by:*

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

where x_{n+1} is the next approximation of the critical point, x_n is the current approximation, $f'(x_n)$ is the first derivative of the function at x_n and $f''(x_n)$ is the second order derivative (Hessian) of the function at x_n.

Newton’s Method Algorithm (Unconstrained Optimization)

1. **Initialize:** Choose a starting point x_0.
2. **Compute gradient:** $g_k = \nabla f(x_k)$.
3. **Compute Hessian:** $H_k = \nabla^2 f(x_k)$.
4. **Compute search direction:** $d_k = -H_k^{-1} g_k$.
5. **Update:** $x_{k+1} = x_k + d_k$.
6. **Check convergence:** If $||g_{k+1}|| < \epsilon$, stop; else repeat.

Newton method

Question: Minimize $f(x_1, x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$ by taking the starting Point as $X_1 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$

Sol. To find X_2.

$$[J_1] = \begin{bmatrix} \frac{d^2f}{dx_1^2} ✓ & \frac{d^2f}{dx_1 dx_2} ✓ \\ \frac{d^2f}{dx_2 dx_1} & \frac{d^2f}{dx_2^2} ✓ \end{bmatrix}$$

$$\frac{df}{dx_1} = 1 + 4x_1 + 2x_2, \boxed{\frac{d^2f}{dx_1 dx_2} = 2}$$

$$\boxed{\frac{d^2f}{dx_1^2} = 4}$$

$$\frac{df}{dx_2} = -1 + 2x_1 + 2x_2, \boxed{\frac{d^2f}{dx_2 dx_1} = 2}$$

$$\boxed{\frac{d^2f}{dx_2^2} = 2}$$

$$[J_1] = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}$$

$$[J_1]^{-1} = \frac{1}{4\times2 - 2\times2} \begin{bmatrix} 2 & -2 \\ -2 & 4 \end{bmatrix}$$

$$[J_1]^{-1} = \frac{1}{4} \begin{bmatrix} 2 & -2 \\ -2 & 4 \end{bmatrix} = \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1 \end{bmatrix}$$

$$4 \begin{bmatrix} -2 & 4 \end{bmatrix} \begin{bmatrix} -1/2 & 1 \end{bmatrix}$$

$$g_1 = \begin{bmatrix} df/dx_1 \\ df/dx_2 \end{bmatrix}_{\underline{X_1}} = \begin{Bmatrix} 1+4x_1+2x_2 \\ -1+2x_1+2x_2 \end{Bmatrix} \begin{Bmatrix} 0 \to x_1 \\ 0 \to x_2 \end{Bmatrix} = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix}$$

$$\therefore X_2 = X_1 - [J_1]^{-1} g_1 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} - \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1/2 \times 1 + (-1/2) \times (-1) \\ -1/2 \times 1 + 1 \times (-1) \end{bmatrix}$$

$$X_2 = \begin{Bmatrix} -1 \\ 3/2 \end{Bmatrix} ✓$$

$$g_2 = \begin{Bmatrix} df/dx_1 \\ df/dx_2 \end{Bmatrix}_{X_2} = \begin{Bmatrix} 1+4x_1+2x_2 \\ -1+2x_1+2x_2 \end{Bmatrix} \begin{Bmatrix} -1 \to x_1 \\ 3/2 \to x_2 \end{Bmatrix} \Rightarrow g_2 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

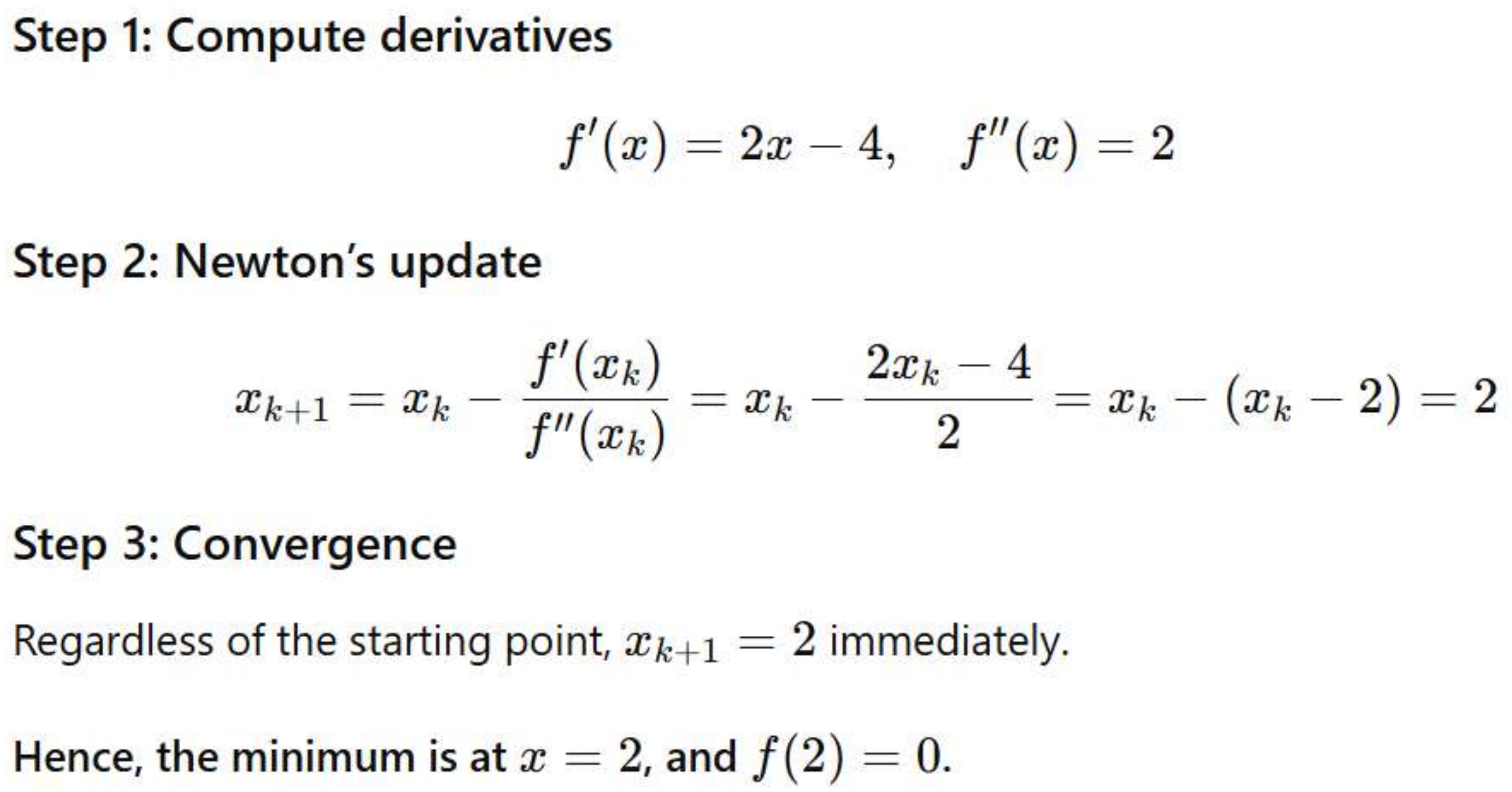
$$X_3 = X_2 - [J_1]^{-1} \cdot g_2$$

$$\begin{bmatrix} \cdot \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\boxed{X_3 = X_2 -} \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$g_1 =$

Example 2 on Newton’s Method – Minimize f(x) = x^2 - 4x +4



Step 1: Compute derivatives

$$f'(x) = 2x - 4, \quad f''(x) = 2$$

Step 2: Newton’s update

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} = x_k - \frac{2x_k - 4}{2} = x_k - (x_k - 2) = 2$$

Step 3: Convergence

Regardless of the starting point, $x_{k+1} = 2$ immediately.

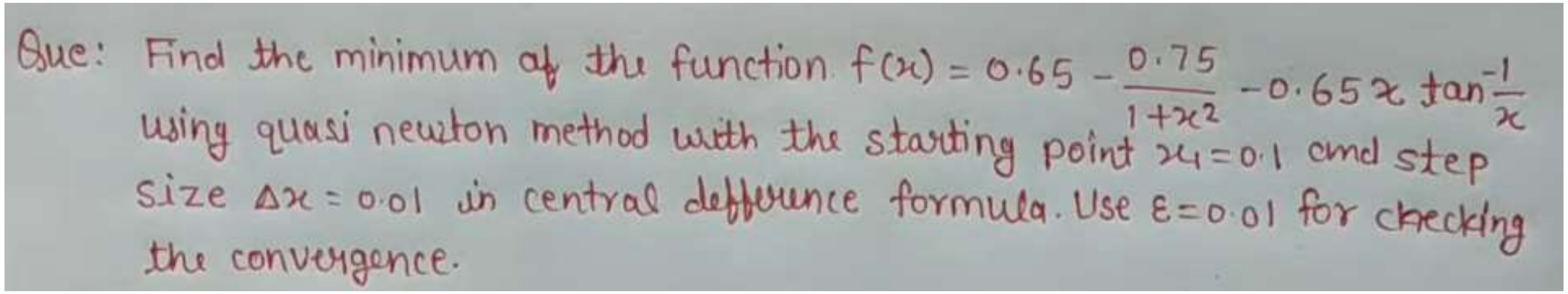
Hence, the minimum is at $x = 2$, and $f(2) = 0$.

Disadvantages of Newton’s Method

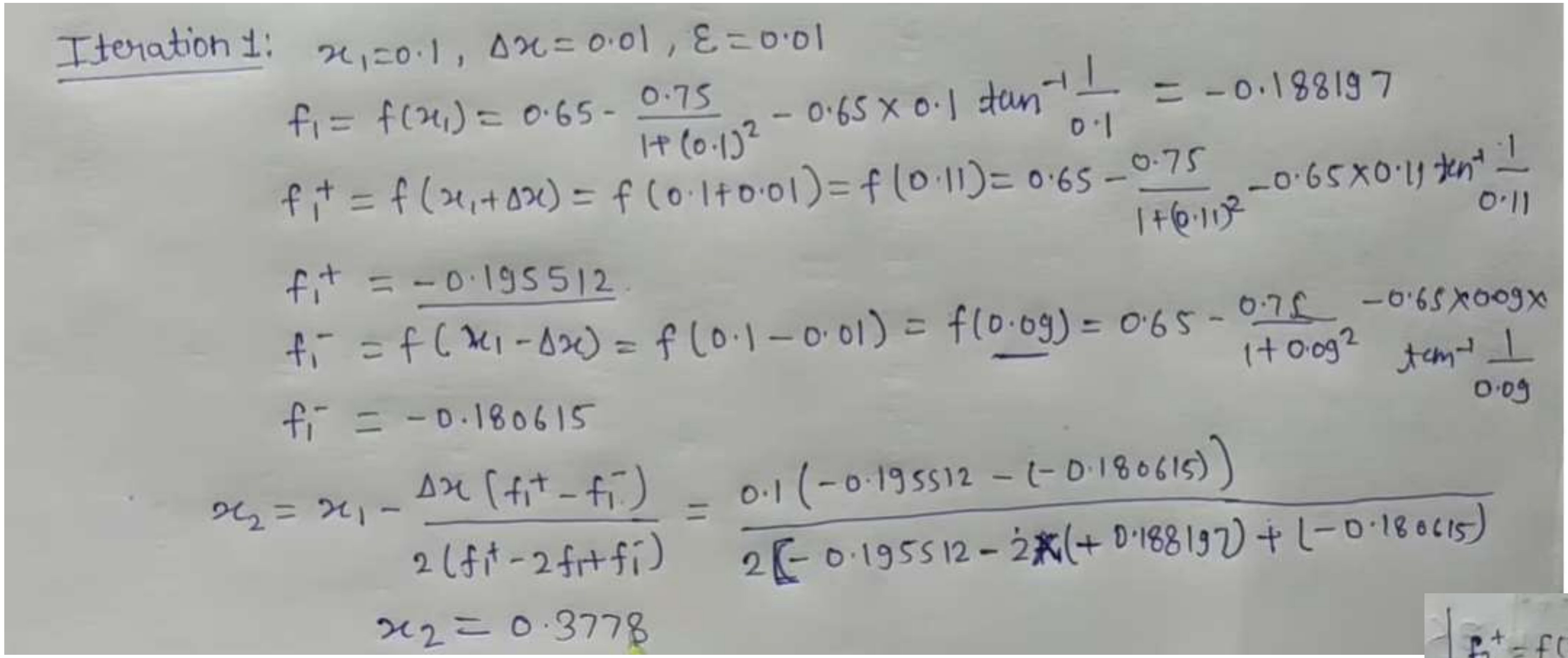
Disadvantage	Explanation
1. Requires Hessian computation	The Hessian ((n \times n) matrix) must be computed and inverted — expensive for large (n).
2. May not converge	If the Hessian is not positive definite (saddle point or maximum), the step can move away from minimum.
3. Sensitive to initial guess	Poor starting point can lead to divergence or convergence to the wrong stationary point.
4. High computational cost	Computing and inverting the Hessian costs (O(n^3)).
5. Not suitable for non-smooth functions	Requires continuous second derivatives.
6. Step may overshoot	If the step size is too large, the quadratic approximation fails — often a line search or damping factor is added.

Quasi Newton Method

- Quasi-Newton methods are optimization algorithms that find minima or maxima of functions by approximating the Hessian matrix of second derivatives.
- Unlike Newton's method, they avoid the computational cost of calculating the true Hessian, instead updating an approximation using gradient and position information from previous steps. This makes them more efficient for large-scale problems while still achieving superlinear convergence.



Que: Find the minimum of the function $f(x) = 0.65 - \frac{0.75}{1+x^2} - 0.65\,x\tan^{-1}\frac{1}{x}$ using quasi newton method with the starting point $x_1 = 0.1$ and step size $\Delta x = 0.01$ in central defference formula. Use $\varepsilon = 0.01$ for checking the convergence.



Iteration 1: $x_1=0.1$, $\Delta x=0.01$, $\varepsilon=0.01$

$$f_1 = f(x_1) = 0.65 - \frac{0.75}{1+(0.1)^2} - 0.65\times 0.1\tan^{-1}\frac{1}{0.1} = -0.188197$$

$$f_1^+ = f(x_1+\Delta x) = f(0.1+0.01) = f(0.11) = 0.65 - \frac{0.75}{1+(0.11)^2} - 0.65\times 0.11\tan^{-1}\frac{1}{0.11}$$

$$f_1^+ = \underline{-0.195512}$$

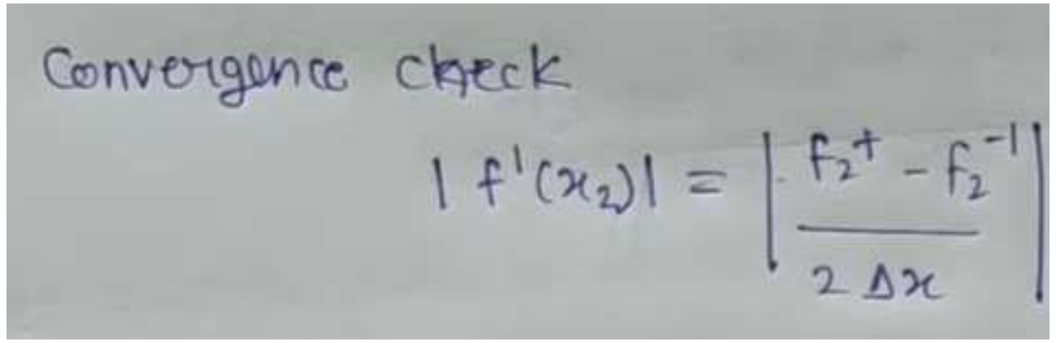
$$f_1^- = f(x_1-\Delta x) = f(0.1-0.01) = f(\underline{0.09}) = 0.65 - \frac{0.75}{1+0.09^2} - 0.65\times 0.09\times\tan^{-1}\frac{1}{0.09}$$

$$f_1^- = -0.180615$$

$$x_2 = x_1 - \frac{\Delta x\,(f_1^+ - f_1^-)}{2(f_1^+ - 2f_1 + f_1^-)} = \frac{0.1(-0.195512-(-0.180615))}{2[-0.195512 - 2(+0.188197) + (-0.180615)}$$

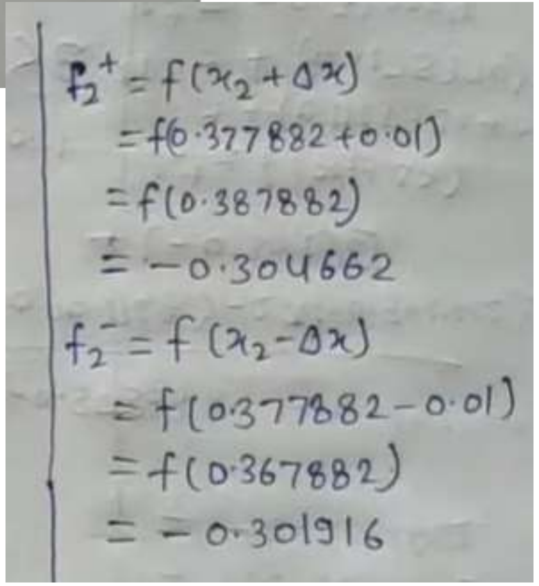
$$x_2 = 0.3778$$

X_2 = 0.377882



Convergence check

$$|f'(x_2)| = \left|\frac{f_2^+ - f_2^{-1}}{2\Delta x}\right|$$



$$f_2^+ = f(x_2+\Delta x)$$
$$= f(0.377882+0.01)$$
$$= f(0.387882)$$
$$= -0.304662$$

$$f_2^- = f(x_2-\Delta x)$$
$$= f(0.377882-0.01)$$
$$= f(0.367882)$$
$$= -0.301916$$

Iteration 2:

$$f_2 = f(x_2) = -0.303368.$$

$$f_2^+ = 0.304662$$

$$f_2^- = -0.301916.$$

$$x_3 = x_2 - \frac{\Delta x(f_2^+ - f_2^-)}{2(f_2^+ - 2f_2 + 2f_2^-)} = 0.465390$$

Convergence check.

$$|f'(x_3)| = \left|\frac{f_3^+ - f_3^-}{2\Delta x}\right| = .$$

$$f_3^+ = f(x_3 + \Delta x)$$
$$= f(0.465390 + 0.01)$$
$$= f(0.475390)$$
$$= -0.310004$$

$$f_3^- = f(x_3 - \Delta x)$$
$$= f(0.465390)$$
$$= -0.45539$$

Convergence check.

$$|f'(x_3)| = \left|\frac{f_3^+ - f_3^-}{2\Delta x}\right| = \underline{0.0177} > \varepsilon = 0.0$$

$$f_3^+ = f(x_3 + \Delta x)$$
$$= f(0.465390 + 0.01)$$
$$= f(0.475390)$$
$$= -0.310004$$

Iteration 3:

$$f_3 = f(x_3) = 0.65 - \frac{0.75}{1+0.46539^2} - 0.65 \times 0.46539 \times \tan^{-1}\frac{1}{0.46539}$$

$f_3 = -0.309885$

$f_3^+ = -0.31004, \quad f_3^- = -0.309650$

$$x_4 = x_3 - \frac{\Delta x(f_3^+ - f_3^-)}{2(f_3^+ - 2f_3 + f_3^-)} = 0.480600$$

Convergence check.

$$|f'(x_4)| = \left|\frac{f_4^+ - f_4^-}{2\Delta\lambda}\right| = 0.000350 < \varepsilon = 0.01$$

$f_4^+ = f(x_4 + \Delta x)$
$= f(0.490600)$
$= -0.3099688$
$f_4^- = f(x_4 - \Delta x)$
$= f(0.47060)$
$= -0.3099615$

. process has converged we take the optimum solution as $x^* \approx x_4 = 0.480600$.

Example: K-Means clustering (Solved Problem)

Data points: (2,10), (2,5), (8,4), (5,8), (7,5), (6,4)

Let K=2 , initial centroids = (2,10) and (5,8)

Data points (2D):

$$P_1 = (2, 10),\ P_2 = (2, 5),\ P_3 = (8, 4),\ P_4 = (5, 8),\ P_5 = (7, 5),\ P_6 = (6, 4).$$

Number of clusters $K = 2$.

Initial centroids chosen:

$\mu_1^{(0)} = (2, 10)$ and $\mu_2^{(0)} = (5, 8)$.

K-Means repeats: (A) assign each point to nearest centroid, (B) recompute centroids as cluster means, until assignments stop changing.

Iteration 1 — Assignment step (distances to initial centroids)

We use **Euclidean distance** $d(a,b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$. (Only relative comparisons matter; I show squared distances to avoid unnecessary square roots.)

Distances to $\mu_1^{(0)} = (2, 10)$

- $P_1 = (2, 10)$: $d^2 = (2-2)^2 + (10-10)^2 = 0$
- $P_2 = (2, 5)$: $d^2 = (2-2)^2 + (5-10)^2 = 25$
- $P_3 = (8, 4)$: $d^2 = (8-2)^2 + (4-10)^2 = 36 + 36 = 72$
- $P_4 = (5, 8)$: $d^2 = (5-2)^2 + (8-10)^2 = 9 + 4 = 13$
- $P_5 = (7, 5)$: $d^2 = (7-2)^2 + (5-10)^2 = 25 + 25 = 50$
- $P_6 = (6, 4)$: $d^2 = (6-2)^2 + (4-10)^2 = 16 + 36 = 52$

  Distances to $\mu_2^{(0)} = (5, 8)$

  - P_1: $d^2 = (2-5)^2 + (10-8)^2 = 9 + 4 = 13$
  - P_2: $d^2 = (2-5)^2 + (5-8)^2 = 9 + 9 = 18$
  - P_3: $d^2 = (8-5)^2 + (4-8)^2 = 9 + 16 = 25$
  - P_4: $d^2 = (5-5)^2 + (8-8)^2 = 0$
  - P_5: $d^2 = (7-5)^2 + (5-8)^2 = 4 + 9 = 13$
  - P_6: $d^2 = (6-5)^2 + (4-8)^2 = 1 + 16 = 17$

Assign each point to the closer centroid (compare squared distances)

- P_1: to μ_1 (0 vs 13) → **Cluster 1**
- P_2: to μ_1 (25 vs 18) → **Cluster 2** (18 smaller)
- P_3: to μ_2 (72 vs 25) → **Cluster 2**
- P_4: to μ_2 (13 vs 0) → **Cluster 2**
- P_5: to μ_2 (50 vs 13) → **Cluster 2**
- P_6: to μ_2 (52 vs 17) → **Cluster 2**

Resulting clusters after Iteration 1:

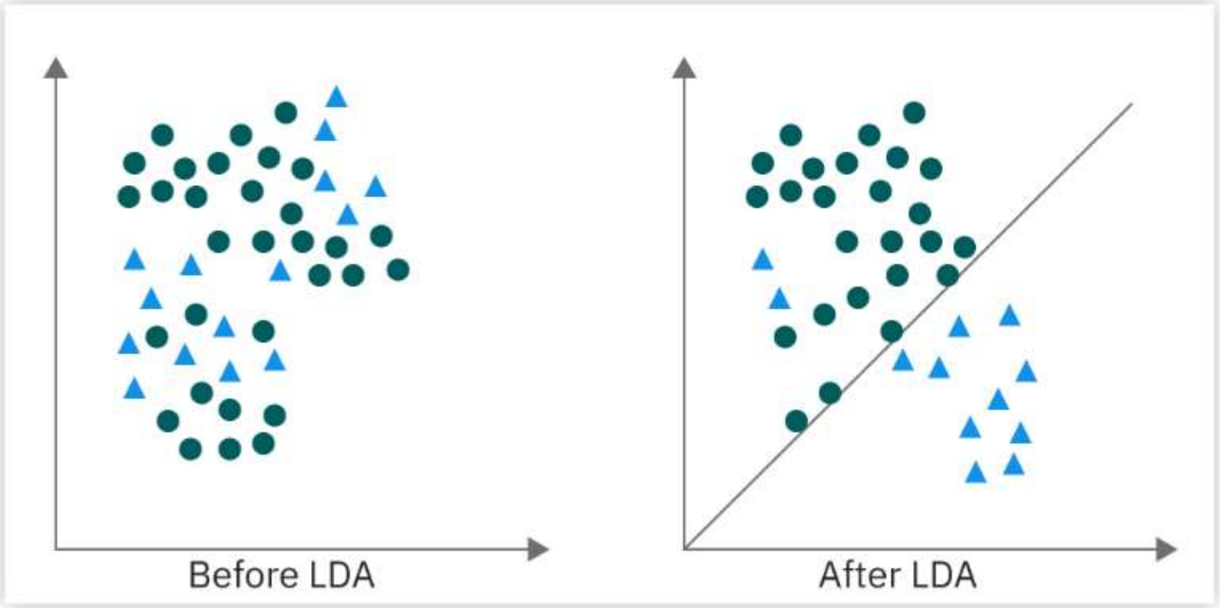
- Cluster 1: $\{P_1\} = \{(2, 10)\}$
- Cluster 2: $\{P_2, P_3, P_4, P_5, P_6\} = \{(2, 5), (8, 4), (5, 8), (7, 5), (6, 4)\}$

Linear Discriminant Analysis(LDA)

- LDA is a supervised learning technique used for both classification and dimensionality reduction.
- Its core idea is to find a linear combination of features that best separates two or more classes of objects or events. It achieves this by maximizing the distance between the means of different classes while minimizing the variation within each class.
- **Key principles:**
  1. **Supervised:** LDA uses labeled data to find the optimal separation between classes, unlike Principal Component Analysis (PCA), which is unsupervised and focuses on maximizing overall data variance..
  2. **Dimensionality Reduction:** LDA projects high-dimensional data onto a lower-dimensional space while retaining the class-discriminatory information. ability. This helps in making subsequent classification tasks more efficient and less prone to overfitting.
  3. **Maximizing Between-Class Variance:** Aims to maximize the separation between the means of different classes.
  4. **Minimizing Within-Class Variance:** Aims to minimize the variance of data points within each class**.**
- Assumptions: LDA assumes that the data within each class follows a Normal distribution and that all classes share the same covariance (variance between classes is same).

Linear Discriminant Analysis(LDA) Applications

- **Classification:** LDA is widely used for multi-class classification problems, particularly when the data is linearly separable or approximately so. Examples include:
  - **Image Recognition:** Face recognition, object classification.
  - **Disease Diagnosis:** Classifying patients into different disease severity levels based on clinical and laboratory data.
  - **Customer Segmentation:** Grouping customers based on their characteristics for targeted marketing.
- **Dimensionality Reduction:**
  - LDA can be used as a preprocessing step to reduce the number of features, which can improve the efficiency and performance of other machine learning algorithms.
- **Feature Extraction:**
  - The linear discriminants found by LDA can be considered as new, more informative features that capture the class separation information.



LDA works by identifying a linear combination of features that separates or characterizes two or more classes of objects or events. LDA does this by projecting data with two or more dimensions into one dimension so that it can be more easily classified.

The technique is, therefore, sometimes referred to as dimensionality reduction. This versatility ensures that LDA can be used for multi-class data classification problems, unlike logistic regression, which is limited to binary classification. LDA is thus often applied to enhance the operation of other learning classification algorithms such as decision tree, random forest or support vector machines (SVM).

Analytic Hierarchy Process(AHP)

- The Analytic Hierarchy Process (AHP) is a structured multi-criteria decision-making (MCDM) method developed by Thomas L. Saaty in the 1970s.
- It helps in making complex decisions by breaking them down into a hierarchy of simpler sub-problems, comparing them, and using mathematical analysis to determine the best alternative.
- **It** is a powerful **multi-criteria optimization and decision-making tool** that helps organizations choose the best alternative by:
  - Structuring the problem,
  - Quantifying subjective preferences,
  - Deriving logical priorities,
  - And ensuring consistency in human judgment.
- Many real-world decisions involve several **conflicting criteria** — for example:
  - Choosing a supplier (based on cost, quality, delivery time, reliability)
  - Selecting a project (based on risk, return, and resources)
  - Deciding on product design or investment alternatives

Scope of AHP

- AHP is widely used to support **rational, transparent, and data-driven decisions** in:
  - **Business decision-making** – project selection, supplier evaluation, investment analysis.
  - **Engineering and manufacturing** – product design, resource allocation, quality control.
  - **Government and policy** – urban planning, transport, and environmental management.
  - **Education and HR** – performance evaluation, recruitment, training needs.
  - **IT and computer science optimization problems** – software selection, system design evaluation.
- **Significance of AHP in Decision Making & Optimization**
  - **Structured Decision-Making:** Converts complex, unstructured problems into hierarchical models.
  - **Combines Qualitative and Quantitative Data:** Incorporates expert judgment and numerical data together.
  - **Priority-Based Optimization:** Determines which criteria or alternative offers maximum benefit.
  - **Checks Consistency:** Ensures logical and rational comparisons.
  - **Supports Group Decisions:** Useful in collaborative environments where multiple stakeholders are involved.
  - **Flexible and Scalable:** Can handle any number of criteria or alternatives.

TOPSIS Method

- **Technique for Order Performance by Similarity to Ideal Solution** (TOPSIS) is a multi-criteria decision-making decision making process which hypothesizes two artificial alternatives- one which has the best level for values of the attributes considered (Positive ideal alternative) and the other one with the worst level for the values of the same (Negative ideal alternative).
- TOPSIS chooses the alternative of shortest the Euclidean distance from the ideal solution and greatest distance from the negative ideal solution. Alternatives are then ranked based on " closest to the ideal solution and farthest from negative ideal solution".
- This method considers 3 three types of attributes–Qualitative benefit attributes, Quantitative benefit attributes and Cost attributes or criteria
- This technique along with AHP (analytical hierarchy process) are widely used in industries for a large number of areas like vendor selection, purchasing of products, selection of logistic providers, network selection, site selection and so on.

TOPSIS (2)

- To make this definition easier, let's suppose you want to buy a mobile phone, you go to a shop and analyze 5 mobile phones on basis of RAM, memory, display size, battery, and price. At last, you're confused after seeing so many factors and don't know how to decide which mobile phone you should purchase.
- TOPSIS is a way to allocate the ranks on basis of the weights and impact of factors.
  - **Weights** mean how much a given factor should be taken into consideration (default weight = 1 for all factors). Like you want RAM to have weighed more than other factors, so the weight of RAM can be 3, while others can have 1.
  - **Impact** means that a given factor has a positive or negative impact. Like you want Battery to be large as possible but the price of the mobile to be less as possible, so you'll assign '+' weight to the battery and '-' weight to the price.
- This method can be applied in ranking machine learning models on basis of various factors like correlation, R-square accuracy, Root mean square error, etc.

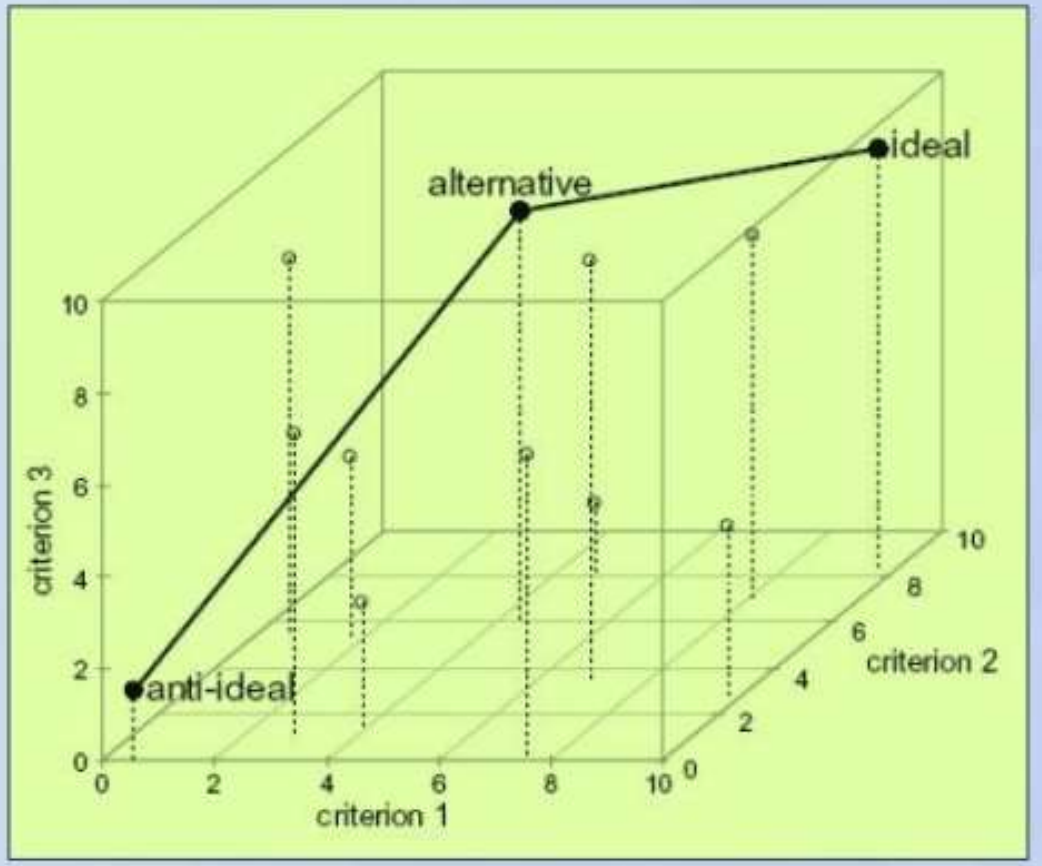
TOPSIS (2)

- To make this definition easier, let's suppose you want to buy a mobile phone, you go to a shop and analyze 5 mobile phones on basis of RAM, memory, display size, battery, and price. At last, you're confused after seeing so many factors and don't know how to decide which mobile phone you should purchase.
- TOPSIS is a way to allocate the ranks on basis of the weights and impact of factors.
  - **Weights** mean how much a given factor should be taken into consideration (default weight = 1 for all factors). Like you want RAM to have weighed more than other factors, so the weight of RAM can be 3, while others can have 1.
  - **Impact** means that a given factor has a positive or negative impact. Like you want Battery to be large as possible but the price of the mobile to be less as possible, so you'll assign '+' weight to the battery and '-' weight to the price.
- This method can be applied in ranking machine learning models on basis of various factors like correlation, R-square accuracy, Root mean square error, etc.

Why TOPSIS?

- Each criterion can be taken into consideration in making a final ranking
  - the concept of TOPSIS is rational
  - the computation involved is easy
  - It allows objective weights to be incorporated into the comparison process.
- TOPSIS method while determining ‘‘ideal” and ‘‘anti-ideal” solutions computes the weighted distances to measure the relative distances away from the ideal and anti-ideal solutions for each alternative (i.e., decision rule).
- Not only the best alternative should be as close as possible to the ideal solution but also ***it should be as far as possible away from the anti-ideal solution***.

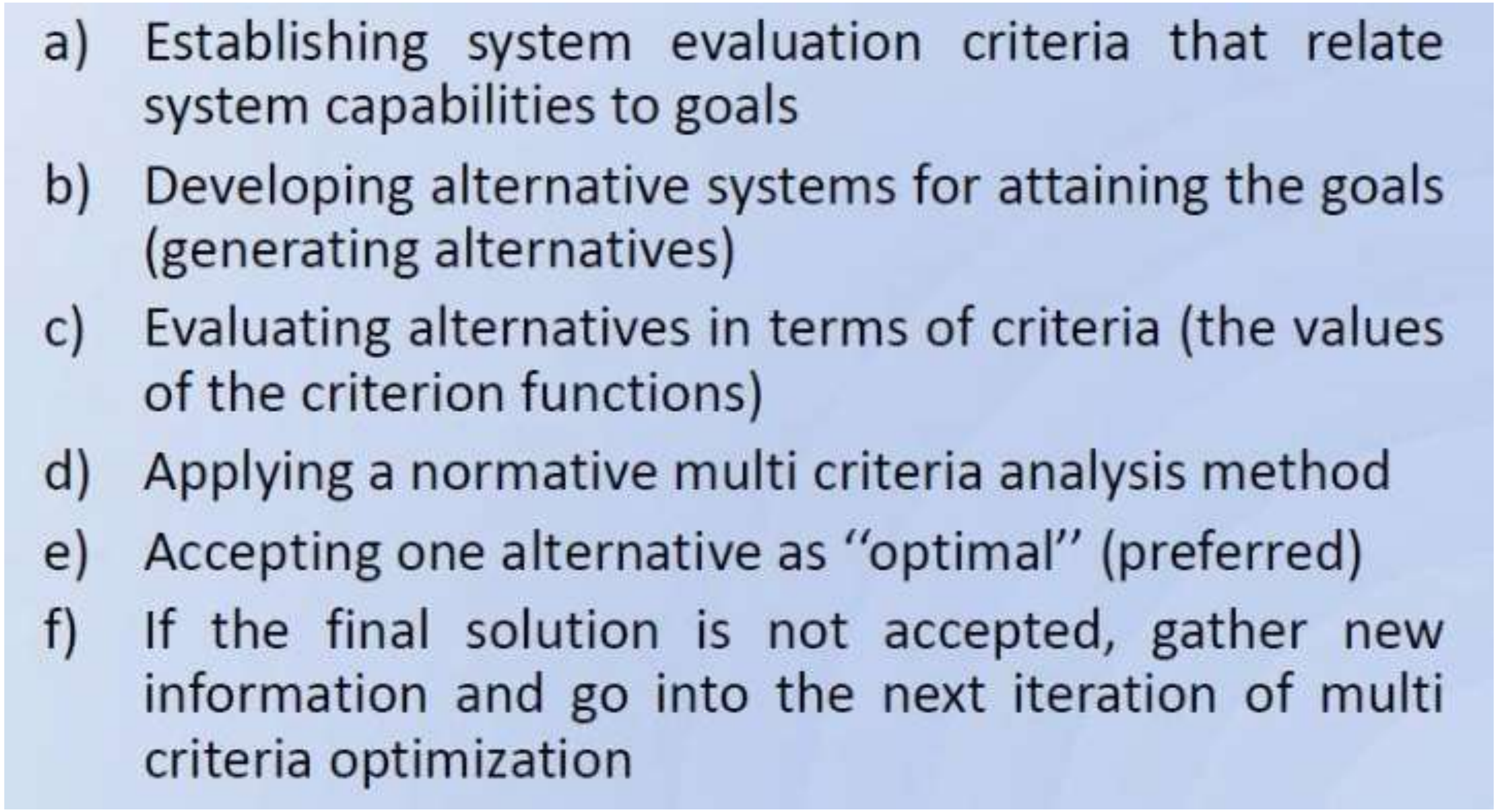
Idea of TOPSIS method



TOPSIS Method(3)

- TOPSIS is thus composed of two components- **Weights and Distances.** The most commonly used weights are Mean weight and standard deviation weight. Among the most commonly used distances is Euclidean distance
- Positive ideal solution is thus defined as a solution which maximizes the benefit criteria and minimizes the cost criteria i.e. the best values attainable from the criteria.
- However Negative ideal solution maximizes the cost criteria and minimizes the benefit criteria i.e. worst values attainable from the criteria.

Steps of TOPSIS Method



a) Establishing system evaluation criteria that relate system capabilities to goals

b) Developing alternative systems for attaining the goals (generating alternatives)

c) Evaluating alternatives in terms of criteria (the values of the criterion functions)

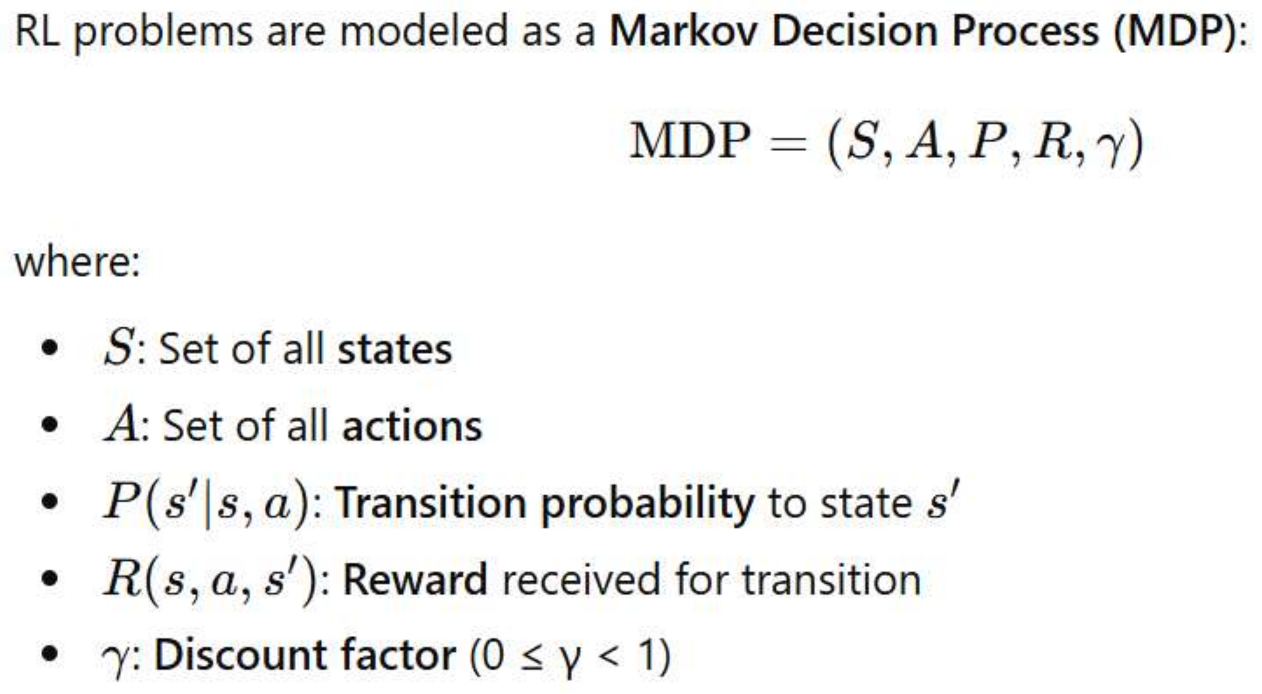
d) Applying a normative multi criteria analysis method

e) Accepting one alternative as "optimal" (preferred)

f) If the final solution is not accepted, gather new information and go into the next iteration of multi criteria optimization

Reinforcement Learning (RL)

- Reinforcement Learning is a branch of machine learning and optimization that focuses on how agents can learn to make decisions by interacting with an **environment** in order to maximize a cumulative reward. Unlike supervised learning (where correct outputs are provided), in RL:
  - The agent **learns through trial and error**.
  - It receives **feedback** in the form of rewards or penalties.
  - It must **balance exploration** (trying new actions) and **exploitation** (using known good actions).
- **Mathematical Representation (based on MDP)**

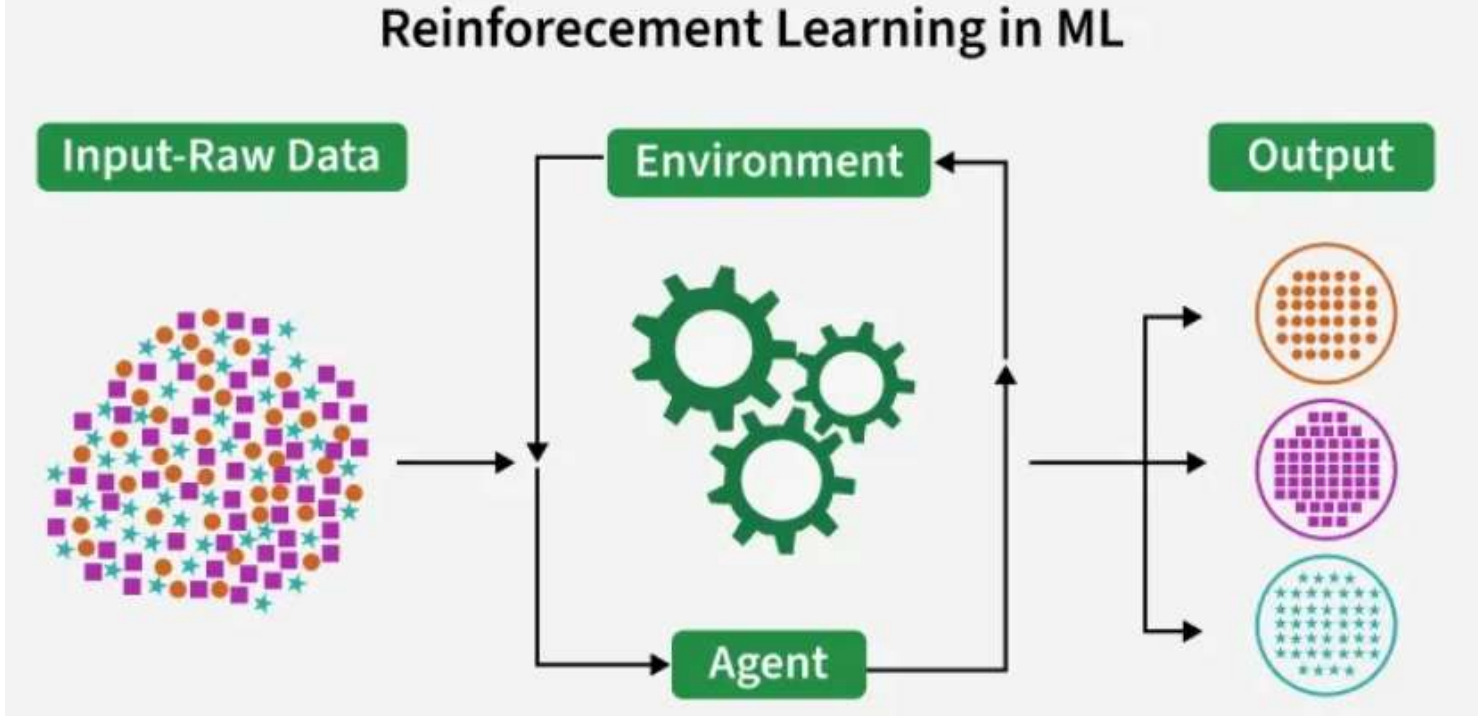


RL problems are modeled as a **Markov Decision Process (MDP)**:

$$\text{MDP} = (S, A, P, R, \gamma)$$

where:

- S: Set of all **states**
- A: Set of all **actions**
- $P(s'|s,a)$: **Transition probability** to state s'
- $R(s,a,s')$: **Reward** received for transition
- γ: **Discount factor** $(0 \leq \gamma < 1)$



RL revolves around the idea that an agent (the learner or decision-maker) interacts with an environment to achieve a goal. The agent performs actions and receives feedback to optimize its decision-making over time.

- **Agent**: The decision-maker that performs actions.
- **Environment**: The world or system in which the agent operates.
- **State**: The situation or condition the agent is currently in.
- **Action**: The possible moves or decisions the agent can make.
- **Reward**: The feedback or result from the environment based on the agent’s action.

Working of Reinforcement Learning (RL)

- The agent interacts iteratively with its environment in a feedback loop:
  1. The agent observes the current state of the environment.
  2. It chooses and performs an action based on its policy.
  3. The environment responds by transitioning to a new state and providing a reward (or penalty).
  4. The agent updates its knowledge (policy, value function) based on the reward received and the new state.
  5. This cycle repeats with the agent balancing exploration (trying new actions) and exploitation (using known good actions) to maximize the cumulative reward over time.
- This process is mathematically framed as a Markov Decision Process_(MDP) where future states depend only on the current state and action, not on the prior sequence of events.

Reinforcement Learning in Optimization Methods

- Reinforcement Learning is deeply rooted in **Optimization Theory**, especially in:

Optimization Concept	Application in RL
Dynamic Programming	Bellman optimality equations, Value Iteration
Stochastic Approximation	Q-learning updates, gradient estimation
Gradient Descent / Ascent	Policy gradient methods
Convex Optimization	Lagrangian duals in constrained RL
Linear Programming	Solving value functions for small MDPs
Quadratic Programming	Support Vector Machines and constrained RL problems
Non-linear Optimization	Deep RL, Neural policy training
Evolutionary Optimization	Genetic algorithms used in RL policy search

Applications of RL in Optimization

1. **Operations Research & Industrial Optimization**
   - Inventory management and supply chain optimization
   - Traffic signal control optimization
   - Scheduling of manufacturing systems
   - **Example:** RL can learn the optimal order quantity or job sequence to minimize total operational cost.
2. **Energy Optimization**
   - Smart grid energy management
   - Power consumption reduction in data centers
   - **Example:** Google DeepMind used RL to optimize cooling systems, reducing energy costs by ~40%.
3. **Financial Optimization**
   - Portfolio management and trading strategy optimization
   - Dynamic pricing models
   - **Example:** RL agents can optimize asset allocations based on reward signals like return or risk ratio.

Applications of RL in Optimization(2)

4. **Robotics and Control Systems**
   - Path planning and navigation
   - Adaptive control in dynamic environments
   - **Example:** Robots learn optimal control actions (torques, angles) to minimize energy use or time.
5. **Machine Learning Optimization**
   - Hyperparameter tuning
   - Reinforcement Learning for feature selection
   - **Example:** RL optimizes learning rates, regularization parameters, or model architectures automatically.
6. **Game Theory and AI**
   - Optimization of strategies in games
   - Example: AlphaGo used deep RL to optimize move selection.

RL in Optimization(3)

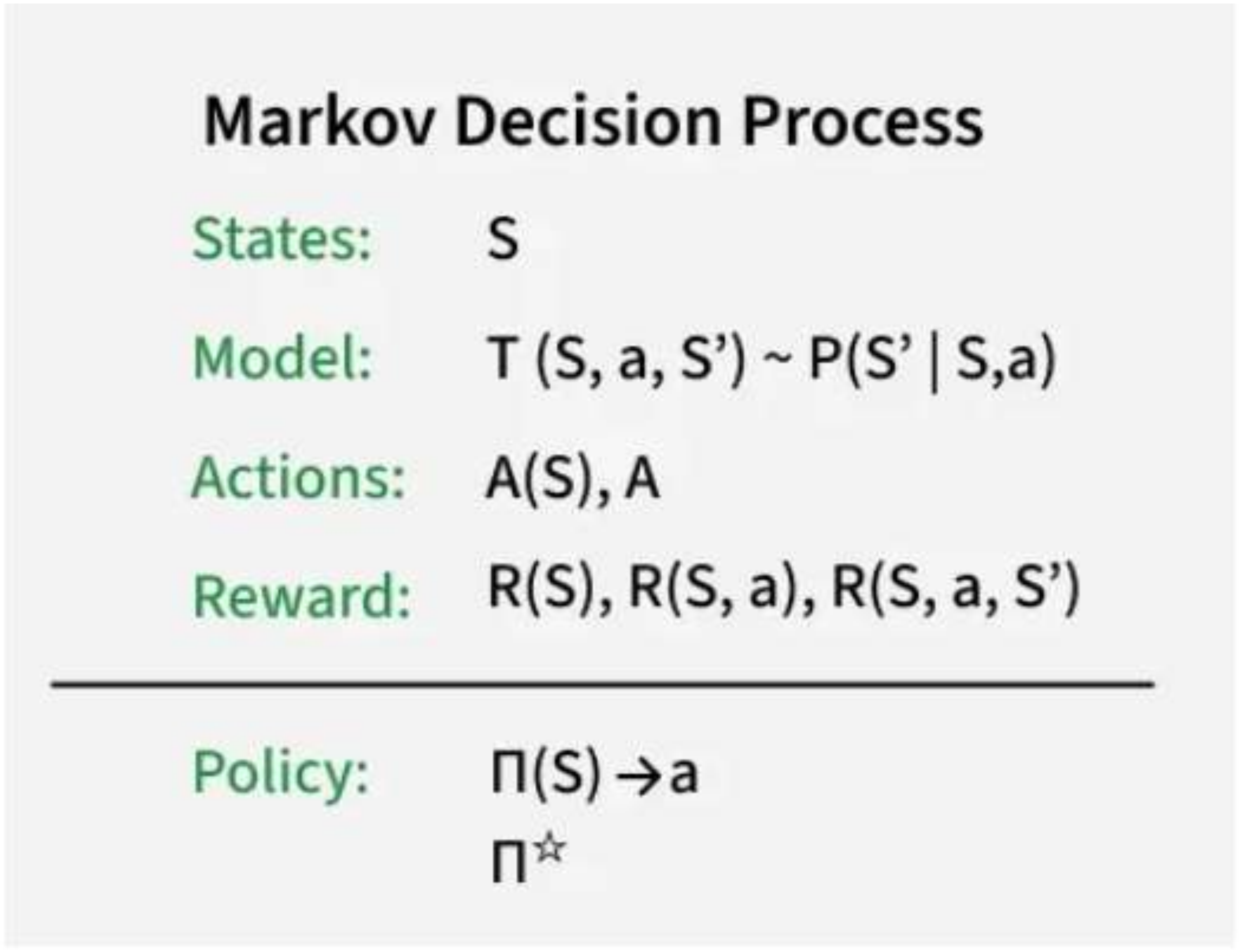
- So, Reinforcement Learning is a **dynamic optimization method** where the objective function (expected cumulative reward) depends on **sequential decisions** and **uncertain outcomes**.
- It integrates **machine learning**, **control theory**, and **optimization** to solve real-world problems involving **long-term, uncertain decision-making**.

Feature	Reinforcement Learning
Learning type	Trial-and-error learning
Feedback	Reward signals
Goal	Maximize cumulative reward
Optimization Type	Sequential, stochastic, dynamic
Core Techniques	Dynamic programming, stochastic optimization, gradient methods
Applications	Robotics, finance, energy, healthcare, operations, games

Markov Decision Process (MDP)

- Markov Decision Process (MDP) is a way to describe how a decision-making agent like a robot or game character moves through different situations while trying to achieve a goal.
- MDPs rely on variables such as the environment, agent's actions and rewards to decide the system's next optimal action. It helps us answer questions like:
    - What actions should the agent take?
    - What happens after an action?
    - Is the result good or bad?
- In AI based applications, MDPs are used to model situations where decisions are made one after another and the results of actions are uncertain. They help in designing smart machines or agents that need to work in environments where each action might led to different outcomes.

Key Components of an MDP



Markov Decision Process

States:	S
Model:	T (S, a, S’) ~ P(S’ | S,a)
Actions:	A(S), A
Reward:	R(S), R(S, a), R(S, a, S’)
Policy:	Π(S) → a $Π^{☆}$

1. States (S): A state is a situation or condition the agent can be in. For example, A position on a grid like being at cell (1,1).

2. Actions (A): An action is something the agent can do. For example, Move UP, DOWN, LEFT or RIGHT. Each state can have one or more possible actions.

3. Transition Model (T): The model tells us what happens when an action is taken in a state. It's like asking: "If I move RIGHT from here, where will I land?" Sometimes the outcome isn't always the same that's uncertainty. For example:

•80% chance of moving in the intended direction

•10% chance of slipping to the left

•10% chance of slipping to the right

This randomness is called a stochastic transition.

4. Reward (R): A reward is a number given to the agent after it takes an action. If the reward is positive, it means the result of the action was good. If the reward is negative it means the outcome was bad or there was a penalty help the agent learn what's good or bad. Examples:

•+1 for reaching the goal

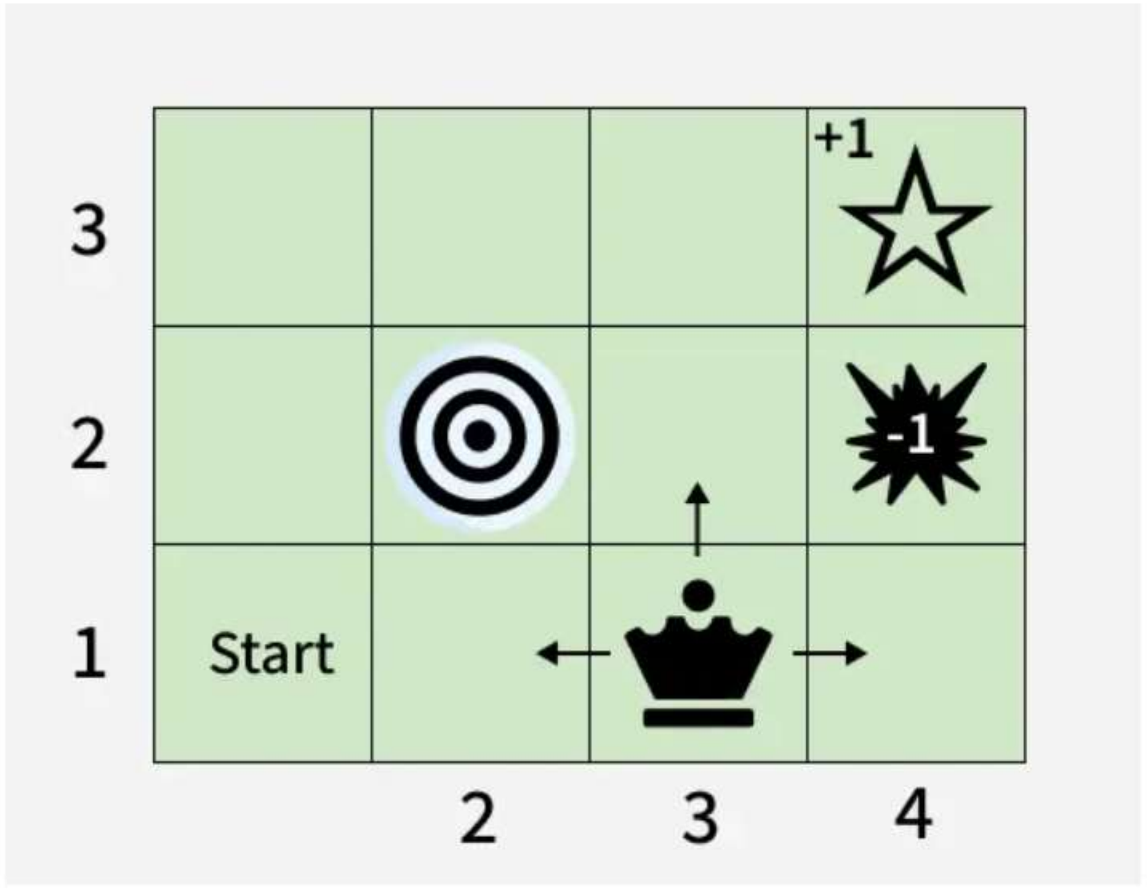
•-1 for stepping into fire

•-0.1 for each step to encourage fewer moves

5. Policy (π): A policy is the agent's plan. It tells the agent: "If you are in this state, take this action." The goal is to find the best policy that helps the agent earn the highest total reward over time.

Example of MDP Problem

Let's consider a 3x4 grid world. The agent starts at cell (1,1) and aims to reach the Blue Diamond at (4,3) while avoiding Fire at (4,2) and a Wall at (2,2). At each state the agent can take one of the following actions: UP, DOWN, LEFT or RIGHT



1. Movement with Uncertainty (Transition Model)

The agent's moves are stochastic (uncertain):
- 80% chance of going in the intended direction.
- 10% chance of going left of the intended direction.
- 10% chance of going right of the intended direction.

2. Reward System

- +1 for reaching the goal.
- -1 for falling into fire.
- -0.04 for each regular move (to encourage shorter paths).
- 0 for hitting a wall (no movement or penalty).

3. Goal and Policy

- The agent's objective is to maximize total rewards.
- It must find an optimal policy: the best action to take in each state to reach the goal quickly while avoiding danger.

4. Path Example

One possible optimal path is: UP → UP → RIGHT → RIGHT → RIGHT
But because of randomness the agent must plan carefully to avoid accidentally slipping into fire.

MDP Applications

- Markov Decision Processes are useful in many real-life situations where decisions must be made step-by-step under uncertainty. Here are some applications:
  - **Robots and Machines:** Robots use MDPs to decide how to move safely and efficiently in places like factories or warehouses and avoid obstacles.
  - **Game Strategy:** In board games or video games MDPs help characters to choose the best moves to win or complete tasks even when outcomes are not certain.
  - **Healthcare:** Doctors can use it to plan treatments for patients, choosing actions that improve health while considering uncertain effects.
  - **Traffic and Navigation:** Self-driving cars or delivery vehicles use it to find safe routes and avoid accidents on unpredictable roads.
  - **Inventory Management:** Stores and warehouses use MDPs to decide when to order more stock so they don’t run out or keep too much even when demand changes.

Logistic Regression

- SUPERVISED LEARNING used in CLASSIFICATION MODEL
- Predict the classes e.g. Predict email is Spam(Y=1) or Not Spam(Y=0)
- **Dependent variable data which is to be precited is categorial and binary (o or 1) in nature.**
- Exam Result to be predicted (Pass Y=1) or Fail (Y = 0) based on the number of study hours.
- There can be one or more independent variables to predict the dependent var. (0 or 1).
- **Logistic Regression** is a **supervised learning** algorithm used for **classification**, not regression and it predicts **categorical outcomes**, usually **binary** (Yes/No, 0/1, Pass/Fail, Spam/Not Spam, etc.).
- Even though the name has "regression," it is actually a **classification algorithm** based on the **logistic (sigmoid) function**.
- Sigmoid function will give values in the range 0 to 1.

Sigmoid

$$\underbrace{y} = \frac{1}{1+e^{-(a_0+a_1 x)}}$$

Independent variable **Dependent variable**

Study Hours	Exam Result
2	0 (Fail)
3	0
4	1 (Pass)
5	1
6	1
7	1
8	1

Logistic Regression

The **sigmoid** function maps any real number to a value between 0 and 1:

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

where

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

Interpretation:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

- $P(Y = 1|X)$ is the **probability** that the outcome is 1 (for example, success, yes, etc.).
- The **decision boundary** is usually set at **0.5**:
  - If $P(Y = 1|X) \geq 0.5 \Rightarrow \text{predict } 1$
  - Else $\text{predict } 0$

Logistic Regression

- SUPERVISED LEARNING used in CLASSIFICATION MODEL
- Algorithm used for **classification**, not regression and it predicts **categorical outcomes of the dependent variable Y**, usually **binary** (Yes/No, 0/1, Pass/Fail, Spam/Not Spam, etc.). Predict the classes e.g. Predict email is Spam(Y=1) or Not Spam(Y=0)
- Exam Result to be predicted (Pass Y=1) or Fail (Y = 0) based on the number of study hours. There can be one or more independent variables to predict the dependent var. (0 or 1).
- Even though the name has "regression," it is actually a **classification algorithm** based on the **logistic (sigmoid) function**. Sigmoid function will give values in the range 0 to 1. here a0 and a1 are based on the MLE (Maximum Likelihood Estimation) method.

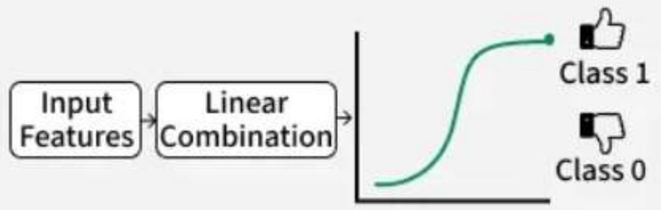
Independent variable **Dependent variable**

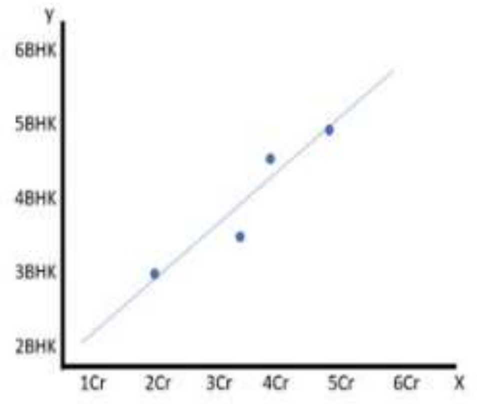
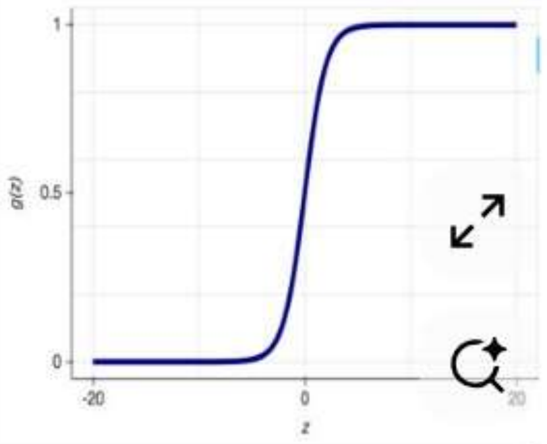
Study Hours	Exam Result
2	0 (Fail)
3	0
4	1 (Pass)
5	1
6	1
7	1
8	1

Sigmoid

$$\underbrace{y} = \frac{1}{1+e^{-(a_0+a_1x)}}$$

- Predicts the probability of a binary outcome (Yes/No, 0/1)
- Uses the sigmoid function to map inputs to probabilities (0 to 1)
- Ideal for classification tasks

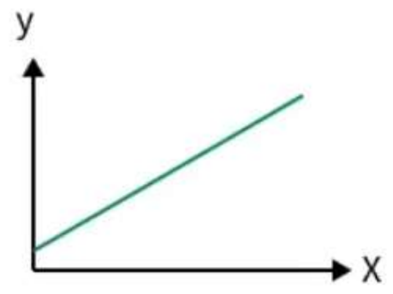


Linear Regression	Logistic Regression
Target is an interval variable	Target is discrete (binary or ordinal) variable
Predicted values are the mean of the target variable at the given values of the input variable	Predicted values are the probability of the particular levels of the given values of the input variable
Solve regression problems	Solve classification problems
Example : What is the Temperature?	Example : Will it rain or not?
Graph is straight line	Graph is S-curve
	

Linear Regression vs Logistic Regression

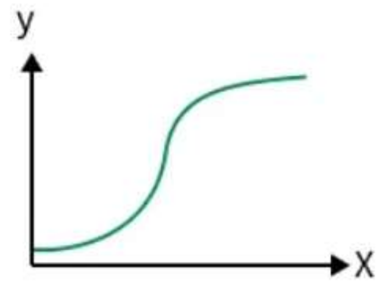
Linear Regression

- Predicts continuous values
- Uses best-fit line
- Solves regression problems



Logistic Regression

- Predicts categorical classes
- Uses sigmoid S-curve
- Solves classification problems



Formula for Logistic Regression Model is :

$$P(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}$$

- $P(Y=1|X)$ is the **probability** that the outcome is 1 (for example, success, yes, etc.).
- The **decision boundary** is usually set at **0.5**:
  - If $P(Y=1|X) \geq 0.5 \Rightarrow \text{predict } 1$
  - Else $\text{predict } 0$

Log-Odds (Logit) formula for Logistic Regression

We can rewrite the logistic model in terms of **log-odds** (or **logit**):

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X$$

This shows that logistic regression models a **linear relationship between the independent variable(s) and the log-odds of the outcome**.

Problem 1

Suppose the logistic regression model is:

$$p = \frac{1}{1 + e^{-(-4+0.8x)}}$$

where x represents **hours studied**, and p is the **probability of passing an exam**.

Find: a) The probability that a student who studied 5 hours passes the exam.

b) The decision (Pass or Fail) if the threshold = 0.5.

SOLUTION : First write the model and in next step substitute x = 5 in the model

$$p = \frac{1}{1 + e^{-(-4+0.8x)}}$$

$$z = -4 + 0.8(5) = -4 + 4 = 0$$

$$p = \frac{1}{1 + e^{-0}} = \frac{1}{1+1} = 0.5$$

Interpretation (Step 3) : The probability of passing when studying 5 hours is **0.5.**

Since the threshold = 0.5, we predict “Pass” if we take p ≥ 0.5

Step 4 : Check another case (for another value of *x*)

If $x = 7$:

$$z = -4 + 0.8(7) = -4 + 5.6 = 1.6$$

$$p = \frac{1}{1 + e^{-1.6}} \approx \frac{1}{1 + 0.201} = 0.832$$

So, the probability of passing when studying 7 hours is **0.83 → Predict Pass.**

If $x = 2$:

$$z = -4 + 0.8(2) = -4 + 1.6 = -2.4$$

$$p = \frac{1}{1 + e^{2.4}} \approx \frac{1}{1 + 11.02} = 0.083$$

Probability of passing is **0.08 → Predict Fail.**

Hours Studied (x)	z	p (Probability of Pass)	Prediction
2	-2.4	0.083	Fail
5	0	0.5	Pass (boundary)
7	1.6	0.832	Pass

Concept	Description
Output	Probability between 0 and 1
Decision rule	Usually threshold = 0.5
Link function	Logit = ln(p / (1 - p))
Estimation	Coefficients (β) are found using **Maximum Likelihood Estimation (MLE)**
Use cases	Binary classification: spam detection, disease prediction, churn prediction, etc.

Multivariate Logistic Regression

When we have **more than one independent variable**, logistic regression generalizes easily.
Model Definition in this case is as follows:

For n features $x_1, x_2, \ldots, x_n$, the logistic regression model is:

$$p = P(Y = 1|X) = \frac{1}{1 + e^{-z}}$$

where

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

Equivalently,

$$\text{logit}(p) = \ln\left(\frac{p}{1 - p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

Example - A company wants to predict whether a customer will buy a product (**Y = 1**) or not (**Y = 0**) based on:

Variable	Description
x_1	Age (in years)
x_2	Monthly Income (in ₹ thousands)

The fitted logistic regression model is:

$$p = \frac{1}{1 + e^{-(-6+0.04x_1+0.3x_2)}}$$

Calculate: a) The probability of purchase for a 30-year-old earning ₹25,000/month.

b) The decision (Buy / Not Buy) at a 0.5 threshold.

Step 1: Compute the value of z

$$z = -6 + 0.04(30) + 0.3(25)$$

$$z = -6 + 1.2 + 7.5 = 2.7$$

4. Step 2: Compute Probability

$$p = \frac{1}{1 + e^{-2.7}} = \frac{1}{1 + 0.067} = 0.937$$

So, **probability = 0.937 (93.7%)** that the customer will buy the product.

5. Step 3: Decision

Since $p = 0.937 > 0.5$,

Prediction: Customer will buy the product (Y = 1).

6. Step 4: Try another case

Customer B: $x_1 = 22$ years, $x_2 = 10$ (₹10,000/month)

$$z = -6 + 0.04(22) + 0.3(10) = -6 + 0.88 + 3 = -2.12$$

$$p = \frac{1}{1 + e^{2.12}} = \frac{1}{1 + 8.33} = 0.107$$

So, $p = 0.107$ → **Prediction: Will not buy (Y = 0).**

7. Step 5: Interpret Coefficients

Coefficient	Meaning
$\beta_0 = -6$	Base log-odds when all predictors = 0.
$\beta_1 = 0.04$	For each **extra year of age**, log-odds of buying increase by 0.04.
$\beta_2 = 0.3$	For each **₹1000 increase in monthly income**, log-odds of buying increase by 0.3.

Thus, **income** has a stronger effect on purchase probability than **age**.

8. Step 6: Decision Boundary

The decision boundary is found when $p = 0.5$, i.e., $z = 0$:

$$-6 + 0.04x_1 + 0.3x_2 = 0$$

$$0.3x_2 = 6 - 0.04x_1$$

$$x_2 = 20 - 0.133x_1$$

So the decision boundary is a **straight line** in the (x_1, x_2) plane dividing "Buy" and "Not Buy" regions.

9. Step 7: Summary Table

Case	Age (x_1)	Income (x_2)	z	p	Decision
A	30	25	2.7	0.937	Buy
B	22	10	-2.12	0.107	Not Buy
C	28	15	-6 + 1.12 + 4.5 = -0.38	0.406	Not Buy
D	35	20	-6 + 1.4 + 6 = 1.4	0.802	Buy

Key Interpretations

•Logistic regression models the **probability** that $Y = 1$as a function of predictors.

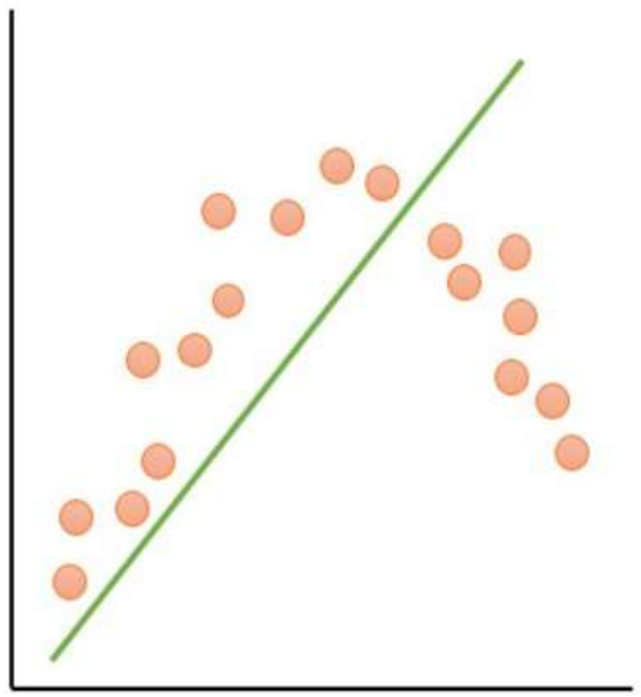
•The coefficients affect the **log-odds**, not directly the probability.

•The model creates a **linear decision boundary** between classes.

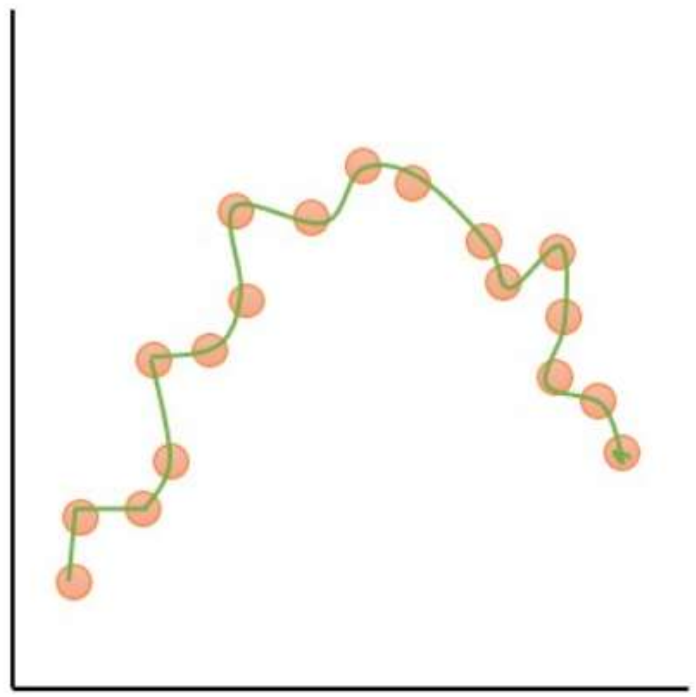
Bias Variance Dichotomy Model (Trade-off Model)

- **Bias** refers to the error that results from oversimplifying the underlying relationship between the input features and the output variable. At the same time, **variance** refers to the error that results from being too sensitive to fluctuations in the training data.
- In Optimization, we strive to **minimize both bias and variance** in order to build a model that can accurately predict on unseen data. A high-bias model may be too simplistic and underfit the training data. In contrast, a model with high variance may overfit the training data and fail to generalize to new data.
- Bias is calculated as the difference between average prediction and actual value. Bias (systematic error) occurs when a model makes incorrect assumptions about data. A model with high bias does not match well training data as well as test data. It leads to high errors in training and test data. While the model with low bias matches the training data well (high training accuracy or less error in training). It leads to low error in training data
- **High Bias** – High bias occurs due to erroneous assumptions in the machine learning model. Models with high bias cannot capture the hidden pattern in the training data. This leads to **underfitting**. Features of high bias are a highly simplified model, underfitting, and high error in training and test data.
- **Low Bias** – Models with low bias can capture the hidden pattern in the training data. Low bias leads to high variance and, eventually, **overfitting**. Low bias generally occurs due to the ML model being overly complex.

High Bias, Underfitting



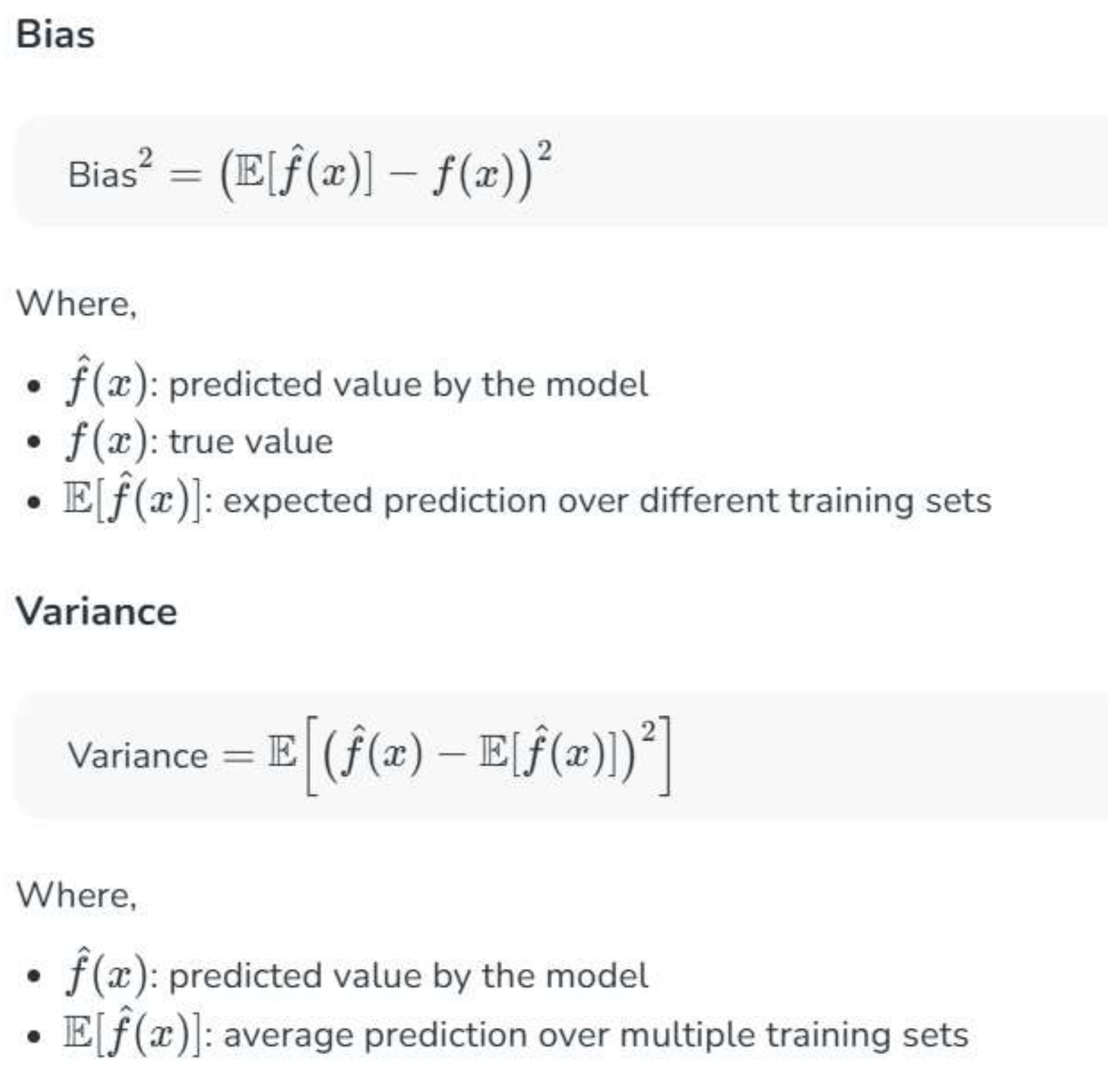
Low Bias, Overfitting



Variance Concept in Bias Variance Dichotomy Model

- **Variance** is a measure of the spread or dispersion of numbers in a given set of observations with respect to the mean.
- In Optimization, Variance is how much a model's predictions change when it's trained on different data.
- It shows how much model prediction varies when there is a slight variation in data. If model accuracies on training and test data vary greatly, the model has high variance.
- A model with high variance can even fit noises on training data but lacks generalization to new, unseen data.
    - **High variance:** The model is too sensitive to small changes and may overfit.
    - **Low variance:** The model is more stable but might miss some patterns

Mathematical Formula for Bias and Variance



Bias

$$\text{Bias}^2 = \left(\mathbb{E}[\hat{f}(x)] - f(x)\right)^2$$

Where,

- $\hat{f}(x)$: predicted value by the model
- $f(x)$: true value
- $\mathbb{E}[\hat{f}(x)]$: expected prediction over different training sets

Variance

$$\text{Variance} = \mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)^2\right]$$

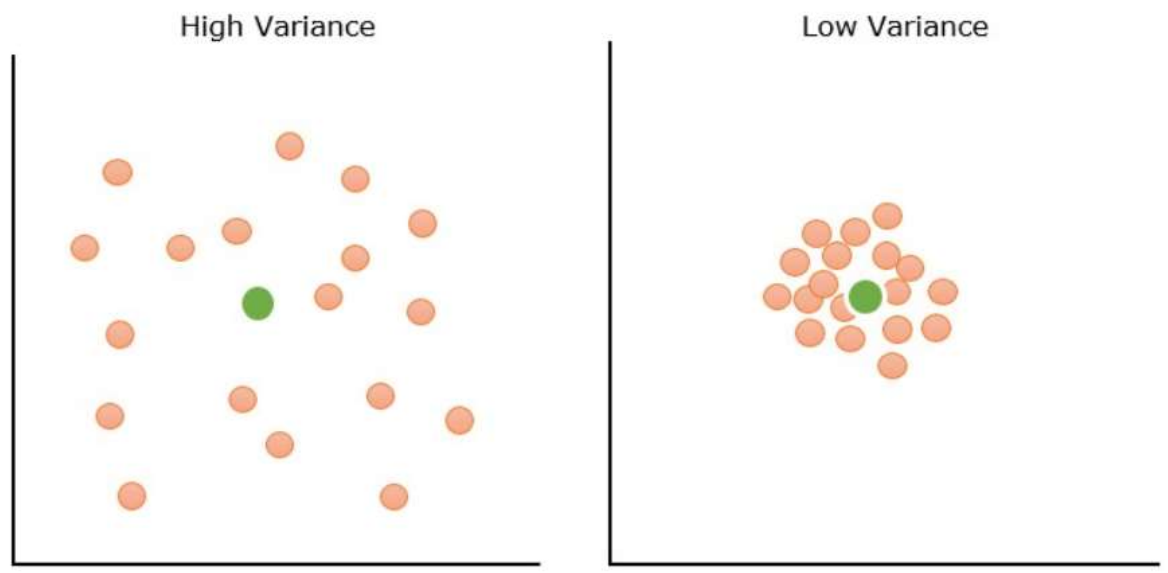
Where,

- $\hat{f}(x)$: predicted value by the model
- $\mathbb{E}[\hat{f}(x)]$: average prediction over multiple training sets

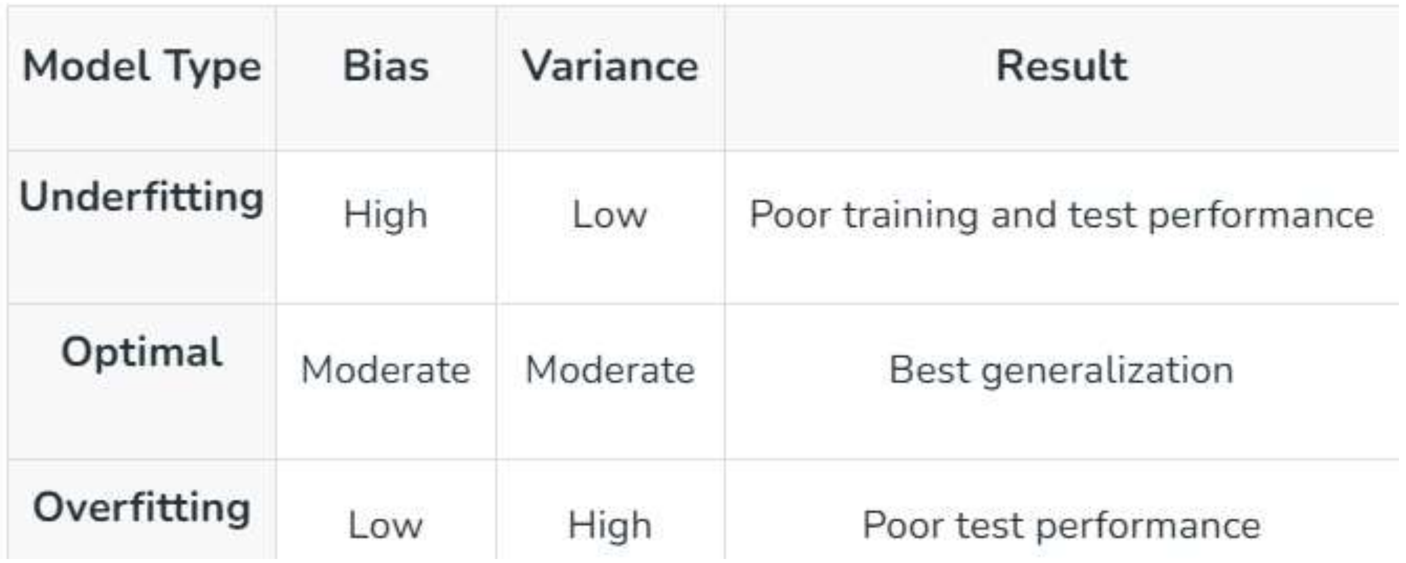
Types of Variance

High Variance – High variance models capture noise along with hidden pattern. It leads to **overfitting**. High variance models show high training accuracy but low test accuracy. Some features of a high variance model are an overly complex model, overfitting, low error on training data, and high error or test data.

Low Variance – A model with low variance is unable to capture the hidden pattern in the data. Low variance may occur when we have a very small amount of data or use a very simplified model. Low variance leads to **underfitting**.



Bias-Variance Tradeoff



Model Type	Bias	Variance	Result
Underfitting	High	Low	Poor training and test performance
Optimal	Moderate	Moderate	Best generalization
Overfitting	Low	High	Poor test performance

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

This decomposition helps us understand why models sometimes **underfit** or **overfit**.

Irreducible Error - This is the noise inherent in data that **no model** can explain.

$$\text{Irreducible Error} = Var(\varepsilon)$$

where ε is the random noise.

Total Expected Prediction Error Formula

The expected mean squared error (MSE) at a point x can be decomposed as:

$$E[(Y - \hat{f}(x))^2] = [\text{Bias}(\hat{f}(x))]^2 + \text{Variance}(\hat{f}(x)) + \sigma^2$$

Where:

- $Y = f(x) + \varepsilon$,
- σ^2 is the variance of noise (irreducible error).

This decomposition is known as the **Bias–Variance Trade-off**.

(a) Irreducible Error

- Comes from the random noise ε.
- Even a perfect model can't predict noise.
- Formally: $\mathrm{Var}(\varepsilon) = \sigma^2$

You **cannot reduce** this part — it's inherent in the data.

(b) Bias

- Bias measures the **systematic error** in your model's assumptions.
- It is the **difference between the true function** $f(x)$ and the **expected prediction** $E[\hat{f}(x)]$ of your model.

$$\mathrm{Bias}(x) = E[\hat{f}(x)] - f(x)$$

and

$$\mathrm{Bias}^2 = [E[\hat{f}(x)] - f(x)]^2$$

High Bias → Model makes strong assumptions, oversimplifies relationships.

Example: Linear regression used for a nonlinear relationship.

(c) Variance

- Variance measures how much $\hat{f}(x)$ would vary if we trained it on different datasets.
- High variance means the model is **too sensitive to training data** — small changes in data cause big changes in prediction.

Formally:

$$\text{Variance}(x) = E[(\hat{f}(x) - E[\hat{f}(x)])^2]$$

High Variance → Model memorizes training data instead of generalizing.

Example: Deep decision trees or k-NN with $k = 1$.

3. Total Error Decomposition

Putting it together:

$$E[(y - \hat{f}(x))^2] = \underbrace{[\text{Bias}(x)]^2}_{\text{Systematic error}} + \underbrace{\text{Variance}(x)}_{\text{Model sensitivity}} + \underbrace{\sigma^2}_{\text{Irreducible noise}}$$

Interpretation

Model Complexity	Bias	Variance	Total Error
Very Simple (Underfit)	High	Low	High
Optimal (Balanced)	Medium	Medium	**Lowest**
Very Complex (Overfit)	Low	High	High

Goal of Model

The learning algorithm aims to **find a balance**:

$$\text{Minimize } (\text{Bias}^2 + \text{Variance})$$

because both extremes lead to high error.

This trade-off guides:

- Model complexity choice
- Regularization techniques (L1, L2)
- Cross-validation strategies
- Ensemble learning methods (bagging reduces variance, boosting reduces bias)

Practical Insight

Situation	Cause	Remedy
High Bias	Model too simple, underfitting	Use more features, increase model capacity
High Variance	Model too complex, overfitting	Regularize, collect more data, use cross-validation

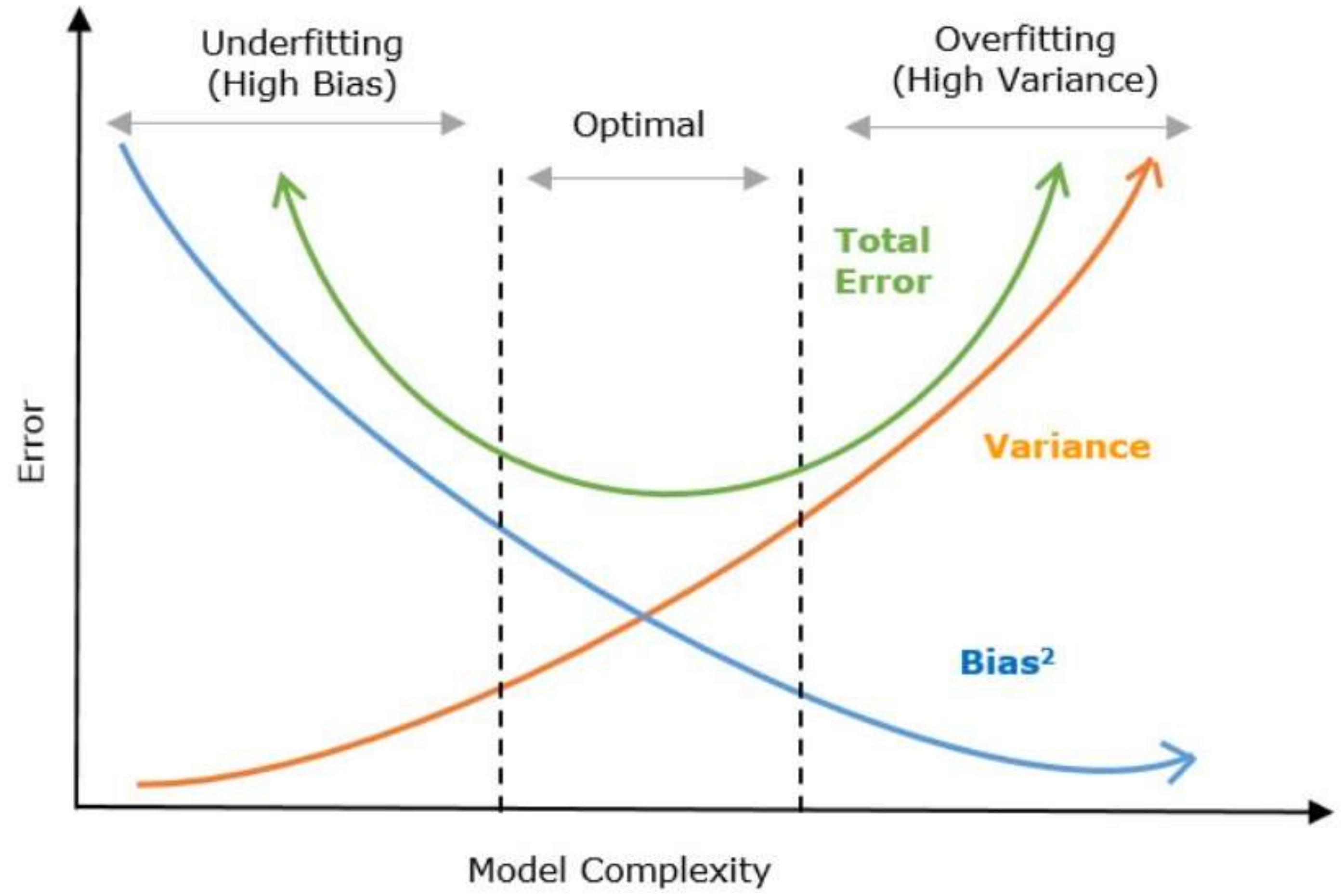
Summary

Concept	Description
Bias	Error from wrong assumptions
Variance	Error from sensitivity to training data
Irreducible Error	Random noise not explainable by model
Goal	Find sweet spot minimizing both Bias^2 and Variance
Techniques to Control Bias/Variance	Regularization (Lasso/Ridge), Cross-validation, Pruning, Bagging/Boosting

The Tradeoff

Model Type	Bias	Variance	Behavior
Simple Model (Linear)	High	Low	Underfits
Complex Model (High-degree polynomial)	Low	High	Overfits

The **goal** is to find an optimal model complexity where the **sum of bias^2 + variance** is minimized.



Example

Suppose we are estimating a function $f(x) = x^2$ using a model trained multiple times on random data.

From several experiments, we observe:

Quantity	Symbol	Value
True value at $x = 2$	$f(2)$	4
Average predicted value $E[\hat{f}(2)]$	3.5	
Average squared prediction $E[\hat{f}(2)^2]$	13.25	

1. **Bias^2**
2. **Variance**
3. **Total Expected Error** (assuming noise variance $\sigma^2 = 0.5$)

Step 1: Compute Bias

$$\text{Bias}(2) = E[\hat{f}(2)] - f(2) = 3.5 - 4 = -0.5$$

$$\text{Bias}^2 = (-0.5)^2 = 0.25$$

Step 2: Compute Variance

$$\text{Variance} = E[\hat{f}(2)^2] - (E[\hat{f}(2)])^2 = 13.25 - (3.5)^2 = 13.25 - 12.25 = 1.0$$

Step 3: Compute Total Expected Error

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

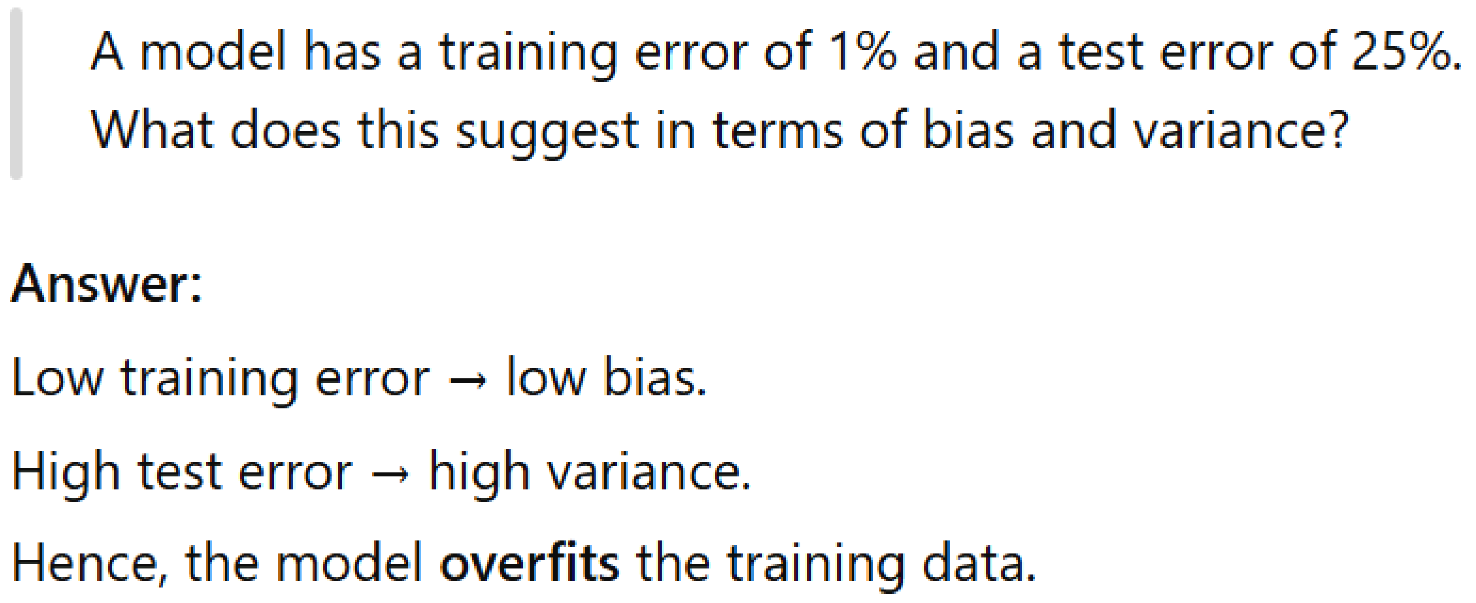
$$= 0.25 + 1.0 + 0.5 = 1.75$$

Interpretation

- **Bias^2 (0.25)** is small → model's predictions are close to the true function.
- **Variance (1.0)** is significant → model predictions vary across datasets.
- **Total error (1.75)** indicates that reducing variance (via regularization or ensemble) could improve model stability.

Summary

Term	Meaning	Desirable?
Bias	Systematic error	Low
Variance	Sensitivity to training data	Low
Irreducible Error	Noise	Unavoidable
Tradeoff	Balance between bias^2 and variance	Optimal complexity minimizes total error



> A model has a training error of 1% and a test error of 25%.
> What does this suggest in terms of bias and variance?

Answer:

Low training error → low bias.

High test error → high variance.

Hence, the model **overfits** the training data.

1. Linear Regression

- It is the most commonly used regression model in machine learning. It may be defined as the statistical model that analyzes the linear relationship between a dependent variable with a given set of independent variables.
- A linear relationship between variables means that when the value of one or more independent variables changes (increase or decrease), the value of the dependent variable will also change accordingly (increase or decrease).
- Linear regression is further divided into two subcategories: simple linear regression and multiple linear regression (also known as multivariate linear regression).
- In simple linear regression, a single independent variable (or predictor) is used to predict the dependent variable. Mathematically, the simple linear regression can be represented as follows- $Y = a + bX$ where,
  - Y is the dependent variable we are trying to predict.
  - X is the independent variable we are using to make predictions
  - b is the slope of the regression line, which represents the effect X has on Y.
  - a is a constant known as the Y-intercept. If $X = 0$, Y would be equal to a.
- In multi-linear regression, multiple independent variables are used to predict the dependent variables.

Multiple Linear Regression Model

- Multiple Linear Regression extends this concept by modelling the relationship between a dependent variable and two or more independent variables. This technique allows us to understand how multiple features collectively affect the outcomes.
- Steps to perform this are similar to that of simple linear Regression but difference comes in the evaluation process. We can use it to find out which factor has the highest influence on the predicted output and how different variables are related to each other. Assumptions of this Model are:
  1. **Linearity**: Relationship between dependent and independent variables should be linear.
  2. **Homoscedasticity**: Variance of errors should remain constant across all levels of independent variables.
  3. **Multivariate Normality**: Residuals should follow a normal distribution.
  4. **No Multicollinearity**: Independent variables should not be highly correlated

- Equation for multiple linear regression is:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

Where:

- $y$is the dependent variable
- $X_1, X_2, \cdots X_n$ are the independent variables
- β_0 is the intercept
- $\beta_1, \beta_2, \cdots \beta_n$ are the slopes

Multicollinearity in Regression Analysis

- Multicollinearity occurs when two or more independent variables in a regression model are highly correlated with each other. So, **multicollinearity e**xists when there are linear relationships among the independent variables, this causes issues in regression analysis because it does not follow the assumption of independence among predictors.
- **Causes of Multicollinearity in Regression Analysis**

  1. **Correlation Among Predictor Variables**: Multicollinearity often occurs when predictor variables in a regression model exhibit a **high correlation** with one another. This situation arises when one predictor variable can be accurately predicted from the others, complicating the estimation of individual predictor effects within the model.
  2. **Overparameterization of the Model**: Introducing too many predictor variables closer to the number of observations can also lead to multicollinearity. More predictors can cause redundancy and increase the variance of the coefficient estimates.
  3. **Data Collection Issues**: Problems in the data collection process can also introduce multicollinearity. For instance, if certain variables are measured with exceptional precision or are inherently interconnected, it can lead to multicollinearity in the regression model.

- **To detect multicollinearity we can use:**

  1. **Correlation Matrix:** A correlation matrix helps to find relationships between independent variables. High correlations (close to 1 or -1) suggest multicollinearity.
  2. **VIF (Variance Inflation Factor):** VIF quantifies how much the variance of a regression coefficient increases if predictors are correlated. A high VIF typically above 10 indicates multicollinearity.

This Concludes Today’s Presentation

Thank you for your attention