Batch Gradient Descent (BGD) – Concept & Math

1. Introduction

- Gradient Descent (GD) is an **optimization algorithm** used to **minimize a cost function** (prediction error) by iteratively updating model parameters.
- **Batch Gradient Descent (BGD)** is one type of GD where **all training data points** are used to calculate the gradient at every step.

2. What is Gradient Descent?

- A method to **find optimal parameters (weights)** for a model by **moving step by step in the direction of steepest descent** of the cost function.

3. Types of Gradient Descent

1. **Batch Gradient Descent (BGD):**
    1. Uses **entire dataset** to compute gradient in each step.
    2. More stable updates but computationally expensive for large datasets.
2. **Stochastic Gradient Descent (SGD):**
    1. Uses **only one sample** per update.
    2. Faster but more noisy and less stable.
3. **Mini-Batch Gradient Descent:**
    1. Uses a **small batch of samples** for each update.
    2. Balances speed and stability.

4. Revision of Gradient Descent

- Formula for updating parameters:

$$\theta_j = \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

Where:

- α = learning rate
- $J(\theta)$ = cost function (e.g., MSE)

5. Mathematical Formulation of BGD

We define:

- **Hypothesis:**

$$\hat{y}_i = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \cdots + \theta_m x_{im}$$

- **Cost Function (Mean Squared Error):**

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

➢ Gradient Calculation

For each parameter θ_j:

$$\frac{\partial J}{\partial \theta_j} = -\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i) x_{ij}$$

➢ Parameter Update Rule

$$\theta_j = \theta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i) x_{ij}$$

- In **Batch GD**, this computation uses **all samples (n)** before a single update.
- Repeat for all parameters until convergence (when J(θ)J(\theta) stops decreasing).

---

6. Characteristics of Batch GD

- ***Advantages:***
  1. More **stable convergence** because full dataset is used.
  2. Accurate direction for moving towards minimum.
- ***Disadvantages:***
  1. Computationally **expensive** for large datasets.
  2. Can be **slow to update parameters** (one update per epoch).

---

7. Creating GDRegressor Class

- Conceptually, this would involve:
  1. Initializing weights.

2. Repeatedly applying the update rule using the **entire dataset**.
3. Stopping when cost function converges or maximum iterations are reached.

➢ Key Takeaways

Term	Meaning
Batch GD	Uses **all samples** to compute gradient per update
Update Rule	$\theta_j = \theta_j - \alpha \frac{1}{n} \sum (y_i - \hat{y}_i) x_{ij}$
Learning Rate (α)	Controls step size towards minimum
Convergence	When cost function stops decreasing significantly
Limitation	Slow for huge datasets (computes gradient over all data every time)
