Gradient Descent – End-to-End Understanding

---

1. Introduction

- Gradient Descent (GD) is one of the **most important optimization algorithms** in machine learning.
- It is used to **find the parameters (weights)** of a model that **minimize the cost (loss) function**, which measures prediction error.

---

2. Summary of Gradient Descent

- GD is an **iterative method**:
    1. Start with **initial values** for the model parameters (randomly chosen).
    2. Calculate how much the cost changes (**gradient**) for each parameter.
    3. Update parameters **step by step** in the direction that **reduces the cost**.
    4. Repeat until we reach the **minimum error point** (optimal solution).

---

3. What is Gradient Descent?

- Gradient Descent is like **rolling down a hill** to reach the **lowest point**.
- The **cost function (J)** is like the hill's surface, where height = prediction error.
- Parameters θ\theta are adjusted step by step to **go downhill** toward the minimum cost.

---

4. Plan of Attack

We aim to:

- Understand the **intuition**.
- Derive the **mathematics**.
- See **how learning rate, data, and loss function** affect GD.

---

5. Intuition for GD

Imagine:

- You are **blindfolded on a hill** (unknown cost surface).
- You can only feel the slope under your feet (gradient).
- You take **small steps downward**, repeating this until reaching the valley (minimum error).

Key terms:

- **Gradient:** Direction of **steepest ascent** of cost function.
- **Negative gradient:** Direction to **minimize cost**.

- **Learning rate (α):** Size of each step toward minimum.

---

6. Mathematical Formulation of Gradient Descent

We have:

- **Cost function:**

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

Where:

$$\hat{y}_i = \theta_0 + \theta_1 x_i$$

We want to **update parameters** $\theta_0, \theta_1, \ldots, \theta_m$.

Step 1: Compute Gradients

For each parameter θj:

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

This gives the **direction and rate of change** of the cost with respect to parameter θj.

Step 2: Update Parameters

$$\theta_j = \theta_j - \alpha\frac{\partial J}{\partial \theta_j}$$

Where:

- α\alpha = learning rate (controls step size).
- Repeat until parameters converge (cost no longer decreases).

---

7. Effect of Learning Rate

- **Too small:** Convergence is **very slow**.
- **Too large:** May **overshoot** the minimum or even **diverge**.
- Optimal learning rate allows **fast, stable convergence**.

---

8. Universality of GD

- GD is **not just for linear regression**:
    1. Works for **neural networks, logistic regression, SVMs**, and many other algorithms.
- As long as you can compute:

1. A **cost function** to minimize.
2. Its **gradient (derivatives)**.

---

9. Effect of Loss Function

- GD depends on the **shape of the cost function**:
    1. **Convex function:** Single global minimum → GD finds it easily.
    2. **Non-convex function:** Multiple local minima → GD may get stuck in one.

Different loss functions lead to different gradient behaviors.

---

10. Effect of Data

- **Scaled data** (normalized features) → Faster convergence.
- **Noisy data** → GD may fluctuate, requiring smaller learning rates.
- **Large datasets** → May use **Stochastic or Mini-batch GD** for efficiency.

➢ Key Takeaways

Term	Meaning
Gradient	Direction of steepest increase of cost
Learning rate (α\alpha)	Controls how big the parameter updates are
Update rule	$\theta_j = \theta_j - \alpha \frac{\partial J}{\partial \theta_j}$
Convergence	Reached when cost no longer decreases
Variants	Batch GD, Stochastic GD, Mini-batch GD
