Logistic Regression Gradient Descent

1. Gradient Descent

- **What it is:**
  An optimization algorithm used to find the parameters (weights) that minimize the loss function.
- **Idea:**
  1. Start with random weights.
  2. Calculate the loss (error) between predictions and actual labels.
  3. Update weights in the opposite direction of the gradient (slope) to reduce error.
- **Update Rule:**

$$w_{\text{new}} = w_{\text{old}} - \alpha \cdot \frac{\partial L}{\partial w}$$

where **α** is the learning rate.

- **In Logistic Regression:**
  The gradient is computed from the derivative of the **Binary Cross-Entropy loss** with respect to weights, which depends on the difference between predicted probabilities and actual labels.

2. Loss Function in Matrix Form

- **Why matrix form?**
  1. Makes calculations efficient for multiple data points using vectorized operations (fast, less error-prone).
- **Matrix Loss Formula:**
  For **m** samples:

$$L = -\frac{1}{m}\left[y^T \log(\hat{y}) + (1-y)^T \log(1-\hat{y})\right]$$

where:

1. y = vector of actual labels (0 or 1)
2. y^ = vector of predicted probabilities from the sigmoid function.

3. Code Demo

Typical steps when coding Logistic Regression from scratch:

1. **Initialize** weights and bias to zeros (or small random values).
2. **Forward pass:**

1. Compute ***z = X w + b***
3. Apply sigmoid: ***y^=σ(z)***
4. **Loss calculation:** Using Binary Cross-Entropy in matrix form.
5. **Backward pass:**
    1. Compute gradient for weights:

$$\frac{\partial L}{\partial w} = \frac{1}{m} X^T(\hat{y} - y)$$

    2. Compute gradient for bias:

$$\frac{\partial L}{\partial b} = \frac{1}{m} \sum(\hat{y} - y)$$

6. **Parameter update:** Apply gradient descent update rule for ww and bb.
7. **Repeat** for several iterations until loss converges.

➢ Key takeaway:

Gradient Descent iteratively tweaks weights to make predictions closer to the truth. In Logistic Regression, this process uses the sigmoid output, Binary Cross-Entropy loss, and vectorized math for speed.
