Gradient Descent

1 Introduction

Gradient descent is an iterative optimization algorithm used to find the minimum of a differentiable function.

2 Algorithm Description

The gradient descent algorithm iteratively updates the parameter x to minimize the function $f(x)$. The update rule is:

$$x_{t+1} = x_t - \eta \cdot f'(x_t),$$

where:

- x_t: Current value of the parameter at iteration t.
- η: Learning rate, controlling the step size.
- $f'(x_t)$: Gradient (derivative) of the function at x_t.

3 Implementation

The Python implementation is as follows:

1. **Function Definition**: Define $f(x) = x^2 + 5x + 6$.
2. **Derivative**: Compute $f'(x) = 2x + 5$.
3. **Gradient Descent**:
   - Initialize $x = x_0$ (e.g., $x_0 = 10$).
   - For each iteration (up to 50 iterations):
     - Compute the gradient: $f'(x) = 2x + 5$.
     - Update x: $x \leftarrow x - \eta \cdot f'(x)$, where $\eta = 0.1$.
     - Store x in a history list for plotting.
   - Return the final x and the history of x values.
4. **Visualization**: Plot $f(x)$ and the path of x values over iterations.

The parameters used are:

- Initial point: $x_0 = 10$.
- Learning rate: $\eta = 0.1$.
- Number of iterations: 50.

4 Analysis of Learning Rate Effects

The learning rate η significantly affects the convergence of gradient descent. We analyze its impact for the function $f(x) = x^2 + 5x + 6$, which has a global minimum at $x = -2.5$ (since $f'(x) = 2x + 5 = 0 \implies x = -2.5$).

4.1 Small Learning Rate ($\eta = 0.01$)

A small learning rate results in small steps toward the minimum.

- **Advantages**:
  - Stable convergence, reducing the risk of overshooting the minimum.
  - Smooth trajectory toward $x = -2.5$.
- **Disadvantages**:
  - Slow convergence, requiring many iterations (e.g., hundreds) to reach $x \approx -2.5$.
  - Computationally expensive for large datasets or complex functions.

4.2 Moderate Learning Rate ($\eta = 0.1$)

The implemented value $\eta = 0.1$ balances speed and stability.

- **Behavior**: Starting at $x_0 = 10$, the algorithm converges to $x \approx -2.5$ within 50 iterations, as shown in the plot.
- **Advantages**:
  - Reasonably fast convergence.
  - Stable updates, avoiding oscillations.
- **Disadvantages**:
  - May still require tuning for other functions or initial points.

4.3 Large Learning Rate ($\eta = 0.5$)

A large learning rate increases step sizes.

- **Behavior**: The algorithm may overshoot the minimum, leading to oscillations or divergence. For $\eta = 0.5$, updates like $x \leftarrow x - 0.5 \cdot (2x + 5)$ can produce large jumps, potentially moving x far from -2.5.
- **Advantages**:
  - Faster initial progress toward the minimum.
- **Disadvantages**:
  - Risk of overshooting, causing oscillations or divergence.
  - Unstable for functions with steep gradients.

4.4 Very Large Learning Rate ($\eta = 1.0$)

For $\eta = 1.0$, the algorithm is highly unstable.

- **Behavior**: Updates become $x \leftarrow x - (2x + 5) = -x - 5$, causing x to alternate signs and grow exponentially (e.g., $x_0 = 10 \implies x_1 = -15 \implies x_2 = 10 \implies \ldots$).
- **Outcome**: Divergence, with x values moving away from the minimum.