Understanding Gradient Descent

Muskula Rahul

Introduction to Gradient Descent

Gradient Descent is a cornerstone optimization algorithm used extensively in machine learning and deep learning for minimizing functions and efficiently training models. It achieves this by iteratively adjusting parameters in the direction that reduces a cost function, effectively navigating the function's landscape to find its minima. This article explains the workings of gradient descent, distinguishes between local and global minima, addresses its shortcomings, and explores advanced variants designed to tackle these limitations.

Working of Gradient Descent

Gradient Descent minimizes a function $f(x)$ by iteratively updating the parameter x using the following formula:

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

where:

- x_t: Current position of the parameter at iteration t.
- η: Learning rate (step size).
- $\nabla f(x_t)$: Gradient of the function at x_t.

The negative gradient ($-\nabla f(x_t)$) indicates the direction of steepest descent, while the learning rate controls the magnitude of each step.

Example Function

Consider the function:

$$f(x) = 0.1x^3 - 2\sin(x) + 0.5x^2 - 1$$

The visualization demonstrates the iterative approach of gradient descent toward a **local minimum**. Starting at an initial point (e.g., $x = 4$), the algorithm adjusts x step-by-step, ultimately converging towards a minimum.

Local vs. Global Minima

Local Minimum

A **local minimum** is a point where the function's value is lower than that of its immediate neighbors, but not necessarily the lowest point across the entire function domain.

Global Minimum

A **global minimum** represents the absolute lowest point of the function across its entire domain.

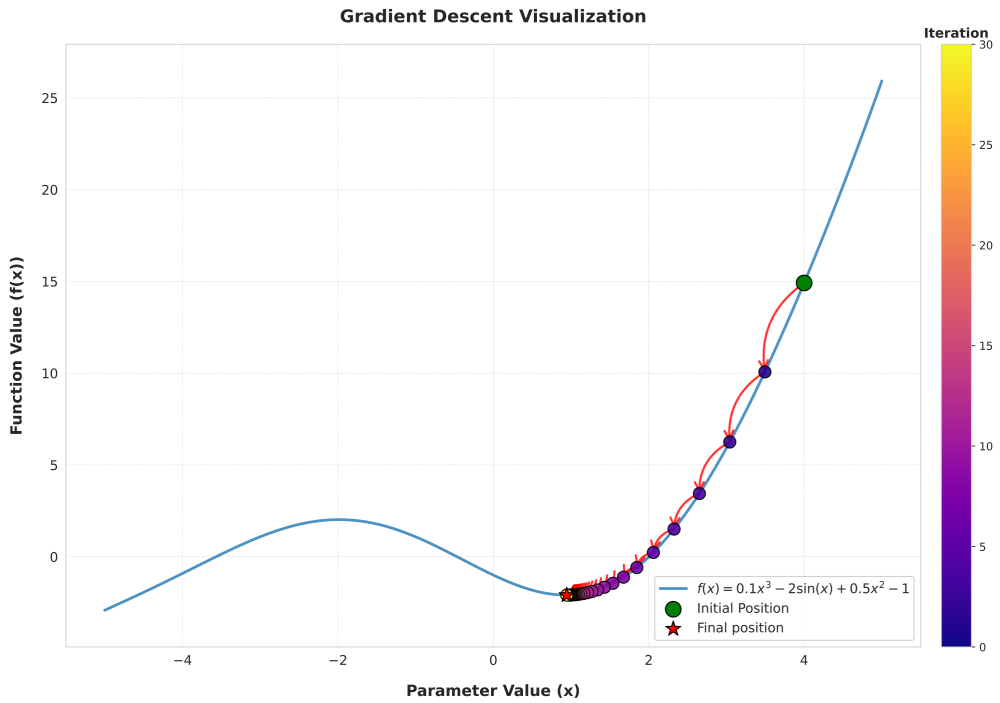

Figure 1: Gradient Descent Visualization. The initial position is marked with a green circle, the final position with a red star. The color of the points represents the iteration number (shown in the colorbar).

Shortcomings of Gradient Descent

1. **Local Minima and Saddle Points:** Gradient descent can become trapped in local minima or saddle points (flat regions where the gradient is near zero).
2. **Learning Rate Sensitivity:** A small learning rate leads to slow convergence, while a large one risks overshooting or divergence.
3. **Non-Convex Functions:** For complex, non-convex functions, the algorithm doesn't guarantee finding the global minimum.
4. **Gradient Vanishing/Exploding:** Gradients can become excessively small (vanish) or large (explode), particularly in deep learning.

Advanced Variants of Gradient Descent

Stochastic Gradient Descent (SGD)

Unlike standard gradient descent, which calculates the gradient using the entire dataset, **Stochastic Gradient Descent (SGD)** updates parameters using a single randomly chosen data point per iteration. This makes it computationally efficient for large datasets.

Update Rule

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta; x_i, y_i)$$

Where:

- θ_t: Current parameters at iteration t.

- η: Learning rate.
- $\nabla_\theta J(\theta; x_i, y_i)$: Gradient for a single data point (x_i, y_i).

Advantages

- Faster updates for large datasets.
- Noise introduced by random sampling can help escape local minima.

Limitations

- Noisy convergence can lead to instability.
- May not converge to the exact minimum.

Mini-Batch Gradient Descent

Mini-Batch Gradient Descent combines standard gradient descent and SGD by using a small subset (mini-batch) of data points for each update.

Update Rule

$$\theta_{t+1} = \theta_t - \eta \frac{1}{|B|} \sum_{i \in B} \nabla_\theta J(\theta; x_i, y_i)$$

Where:

- B: Mini-batch of size $|B|$.
- Other terms are as defined in SGD.

Advantages

- Balances speed and convergence stability.
- Reduces memory usage compared to full-batch gradient descent.

Momentum

Momentum adds a fraction of the previous parameter update to the current update, which accelerates convergence and helps escape local minima.

Update Rules

$$v_t = \beta v_{t-1} + (1 - \beta) \nabla_\theta J(\theta)$$
$$\theta_{t+1} = \theta_t - \eta v_t$$

Where:

- v_t: Velocity (cumulative moving average of gradients).
- β: Momentum coefficient ($0.8 \leq \beta \leq 0.99$).

Advantages

- Accelerates convergence, especially in ravines.
- Helps bypass local minima or saddle points.

RMSprop (Root Mean Square Propagation)

Description

RMSprop adjusts the learning rate dynamically by maintaining a moving average of squared gradients.

Update Rules

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta) g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

Where:

- $E[g^2]_t$: Exponential moving average of squared gradients.
- g_t: Gradient at step t.
- ϵ: Small constant (e.g., 10^{-8}) to prevent division by zero.
- β: Smoothing constant (e.g., 0.9).

Advantages

- Adapts learning rate per parameter.
- Suitable for non-stationary objectives.

Adam (Adaptive Moment Estimation)

Description

Adam combines the benefits of Momentum and RMSprop by maintaining moving averages of both gradients and their squared values.

Update Rules

1. Compute biased moment estimates:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

2. Correct bias:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

3. Update parameters:

$$\theta_{t+1} = \theta_t - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Where:

- m_t, v_t: Biased first and second moment estimates.
- β_1, β_2: Exponential decay rates (0.9 and 0.999).
- ϵ: Small constant (10^{-8}).

Advantages

- Combines the strengths of Momentum and RMSprop.
- Effective for sparse gradients.
- A robust default choice for deep learning tasks.

Cautionary Notes

1. **Local vs. Global Minima:** No guarantee of finding the global minimum for non-convex functions.
2. **Learning Rate Tuning:** Experimentation is crucial. Consider learning rate schedules.
3. **Feature Scaling:** Normalize/standardize input features.
4. **Visualization:** Essential for understanding convergence behavior.

Conclusion

Gradient Descent is a vital tool in optimization. While it has limitations, advanced variants like Adam and RMSprop address many of these challenges. Careful hyperparameter tuning and understanding the trade-offs between local and global minima are key to effective application. Visualization provides invaluable insights.