Gradient Descent and Its Applications in Machine Learning

Rewan Khaled

August 29, 2025

1 Introduction

Gradient descent is an iterative optimization algorithm used to minimize the loss function of a machine learning model. It updates parameters θ in the direction of the negative gradient of the cost function $J(\theta)$:

$$\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

where η is the learning rate. It is widely used in machine learning for optimizing models with multiple parameters.

2 Applications and Variants

Gradient descent is core to many models:

- **Linear Regression:** minimizes mean squared error.
- **Logistic Regression:** minimizes cross-entropy loss.
- **Neural Networks:** backpropagation uses gradient descent.
- **SVM:** optimizes hinge loss for large datasets.

Variants:

- **Stochastic Gradient Descent (SGD):** updates per sample.
- **Mini-batch Gradient Descent:** updates per small batch.
- **Momentum:** accelerates convergence.
- **Adam Optimizer:** combines momentum and adaptive learning rates.

3 Step-by-Step Example and Learning Rate

Suppose data points $(x, y) = \{(1, 2), (2, 3), (3, 5)\}$, initial $\theta = 0$, learning rate $\eta = 0.1$:

1. Compute predictions: $h_\theta(x)$
2. Compute gradient: $\frac{\partial J(\theta)}{\partial \theta}$
3. Update parameters: $\theta := \theta - \eta \cdot \text{gradient}$
4. Repeat until convergence

Learning rate η affects convergence:

- Too small: slow convergence
- Too large: may overshoot or diverge
- Adaptive rates (Adam) can accelerate training

4 Mathematical Derivation

For linear regression with m examples:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

The gradient with respect to θ is:

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)}$$

5 Conclusion and Insights

Gradient descent is a foundational algorithm for optimizing machine learning models. By iteratively adjusting parameters using gradients, models minimize error efficiently.

Advantages: simple, widely applicable, efficient, works with differentiable cost functions. **Limitations:** sensitive to learning rate, can get stuck in local minima, requires differentiable functions.

Understanding variants, step-by-step updates, and hyperparameter effects ensures effective training of models from linear regression to deep neural networks.