Gradient Descent, Convex Functions, and Linear Regression

Optimization in Machine Learning

Study Notes from January 2016

January 5, 2026

Contents

1 Introduction

This document provides a comprehensive overview of optimization techniques used in machine learning, particularly focusing on gradient descent and its application to linear regression. The material covers fundamental concepts in calculus, convex analysis, and numerical optimization.

2 Convex Functions and Optimization

2.1 Definition of Convexity

A function $J(\theta)$ is **convex** if for any two parameter vectors θ_1, θ_2 and any $\lambda \in [0, 1]$, the following inequality holds:

$$J(\lambda\theta_1 + (1-\lambda)\theta_2) \leq \lambda J(\theta_1) + (1-\lambda)J(\theta_2) \tag{1}$$

This definition states that the function value at any convex combination of two points lies below or on the line segment connecting the function values at those points.

2.2 Key Properties of Convex Functions

- **Single global minimum:** Every local minimum of a convex function is also a global minimum.
- **No local minima:** Convex functions cannot have local minima that are not global minima.
- **Differentiability:** If a convex function is twice-differentiable, its Hessian matrix (matrix of second derivatives) is positive semi-definite everywhere.
- **Optimization guarantee:** Optimization algorithms are guaranteed to find the global minimum for convex functions.

2.3 Convexity and the Hessian

For a twice-differentiable function $J(\theta)$, a sufficient condition for convexity is that the Hessian matrix H is positive semi-definite, meaning all eigenvalues are non-negative:

$$H = \begin{bmatrix} \frac{\partial^2 J}{\partial \theta_0^2} & \frac{\partial^2 J}{\partial \theta_0 \partial \theta_1} & \cdots \\ \frac{\partial^2 J}{\partial \theta_1 \partial \theta_0} & \frac{\partial^2 J}{\partial \theta_1^2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{2}$$

The squared-error loss used in linear regression is convex, ensuring a unique global minimum.

3 Taylor Approximation and Local Behavior

3.1 First-Order Taylor Expansion

For a differentiable function $J(\theta)$ and a small parameter change $\Delta\theta$, the first-order Taylor approximation is:

$$J(\theta + \Delta\theta) \approx J(\theta) + \nabla J(\theta)^T \Delta\theta \tag{3}$$

where $\nabla J(\theta)$ is the gradient vector. This approximation captures the local linear behavior of the function around θ.

3.2 Interpretation

- The gradient $\nabla J(\theta)$ gives the direction of **steepest ascent**.
- To minimize the loss, we must move in the **negative gradient direction**.
- The linear approximation guides gradient-based optimization algorithms.

4 Cost Functions for Linear Regression

4.1 Squared-Error Loss

Given supervised data (x_i, y_i) for $i = 1, 2, \dots, n$, with model predictions $\hat{y}_i = \theta^T x_i$, the **squared-error cost function** is:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^T x_i)^2 \tag{4}$$

4.2 Matrix Form

Using matrix notation with design matrix $X \in \mathbb{R}^{n\times m}$ and target vector $y \in \mathbb{R}^n$:

$$J(\theta) = \frac{1}{n}(X\theta - y)^T(X\theta - y) \tag{5}$$

This reveals the **quadratic structure** of the cost function in terms of the parameters θ.

4.3 Why Square the Error?

- **Penalizes large errors:** Squaring emphasizes larger mistakes more strongly than small ones.
- **Convexity:** The quadratic form ensures a unique global minimum.
- **Mathematical convenience:** The squared error has nice calculus properties for optimization.
- **Statistical interpretation:** Under Gaussian noise assumptions, squared error corresponds to maximum likelihood estimation.

5 Gradient and Partial Derivatives

5.1 Partial Derivatives

For a function $J(\theta_0, \theta_1, \dots, \theta_m)$ depending on multiple parameters, the **partial derivative** with respect to parameter θ_j is:

$$\frac{\partial J}{\partial \theta_j} \tag{6}$$

This measures how sensitive the loss is to changes in θ_j when all other parameters are held constant.

5.2 The Gradient Vector

The **gradient** is the column vector of all partial derivatives:

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \vdots \\ \frac{\partial J}{\partial \theta_m} \end{bmatrix} \tag{7}$$

Key theorem (Multivariable Calculus): The gradient $\nabla J(\theta)$ points in the direction of steepest ascent of J. Therefore, $-\nabla J(\theta)$ points in the direction of steepest descent.

5.3 Gradient of Squared-Error Loss

Starting from:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^T x_i)^2 \tag{8}$$

Differentiating with respect to θ:

$$\frac{\partial J}{\partial \theta} = \frac{2}{n}\sum_{i=1}^{n}(\theta^T x_i - y_i)x_i \tag{9}$$

In matrix notation:

$$\nabla J(\theta) = \frac{2}{n}X^T(X\theta - y) \tag{10}$$

where $(X\theta - y)$ is the vector of prediction errors, and X^T weights each feature by its contribution to the error.

6 Gradient Descent Algorithm

6.1 The Update Rule

Gradient descent is an iterative optimization algorithm that updates parameters in the direction opposite to the gradient:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla J(\theta^{(t)}) \tag{11}$$

Components:

- $\theta^{(t)}$: Parameters at iteration t
- $\nabla J(\theta^{(t)})$: Gradient at current parameters
- $\alpha > 0$: Learning rate (step size)
- $\theta^{(t+1)}$: Updated parameters

6.2 Intuitive Interpretation

Gradient descent performs **iterative downhill walking** on the loss surface. At each step:

1. Compute gradient (direction of steepest increase)
2. Move opposite to gradient (direction of steepest decrease)
3. Repeat until convergence

7 Learning Rate: Theory and Practice

7.1 Role of Learning Rate

The learning rate α controls the step size in each iteration. It is arguably the most important hyperparameter in gradient descent.

$$\text{New parameters} = \text{Current parameters} - \alpha \times \text{Gradient} \tag{12}$$

7.2 Small Learning Rate (α too small)

Consequences:

- Very slow convergence to the minimum
- Many iterations required for acceptable accuracy
- High computational cost
- May be impractical for large datasets

When to use: When loss landscape is noisy or prone to oscillation, small α can ensure stability.

7.3 Large Learning Rate (α too large)

Consequences:

- Algorithm overshoots the minimum
- Parameters oscillate around the optimal solution
- Loss may increase instead of decreasing
- **Divergence:** Algorithm may fail to converge, with loss increasing unboundedly

When to use: Generally avoided unless specifically tuned for the problem.

7.4 Optimal Learning Rate

Characteristics:

- Fast convergence while maintaining stability
- Balanced between speed and accuracy
- Problem-dependent: different datasets need different α
- For convex quadratic problems, optimal α relates to eigenvalues of the Hessian

7.5 Learning Rate Selection Strategy

1. **Start conservative:** Begin with a small α (e.g., 0.01)
2. **Increase gradually:** Increase α while monitoring loss decrease
3. **Watch for divergence:** If loss starts increasing, reduce α
4. **Use validation:** Monitor loss on held-out data to ensure generalization
5. **Adaptive methods:** Consider algorithms like Adam, RMSprop that adjust α during training

8 Convergence Analysis

8.1 Convergence Conditions

For convex functions, gradient descent converges to the global minimum under:

- **Appropriate learning rate:** α must be small enough to prevent divergence
- **Sufficient iterations:** May require many iterations for high accuracy
- **Differentiability:** Function must be differentiable almost everywhere

8.2 Convergence Rate

- **Linear convergence:** Loss decreases geometrically with iteration count
- **Quadratic functions:** For perfectly quadratic loss (ideal case), convergence is fastest
- **Non-smooth regions:** Convergence slows near boundaries or non-smooth areas

8.3 Stopping Criteria

Common approaches to determine when to stop gradient descent:

1. **Maximum iterations:** Stop after T iterations
2. **Gradient norm:** Stop when $\|\nabla J(\theta)\| < \epsilon$ (very small gradient)
3. **Loss improvement:** Stop when loss change between iterations is negligible
4. **Validation-based:** Stop when validation loss stops improving (early stopping)

9 Second-Order Conditions

9.1 Convexity Check via Hessian

For a function to be convex, the Hessian must be positive semidefinite:

$$v^T H v \geq 0 \quad \forall v \in \mathbb{R}^m \tag{13}$$

Equivalently, all eigenvalues of H must be non-negative.

9.2 Squared-Error Hessian

For squared-error loss:

$$H = \frac{2}{n} X^T X \tag{14}$$

Since $X^T X$ is a Gram matrix (always positive semidefinite), squared-error loss is convex, guaranteeing a unique global minimum.

10 Summary and Key Takeaways

- **Convex optimization:** Linear regression has a convex cost function with a unique global minimum.
- **Gradient descent:** Iteratively moves in the negative gradient direction to minimize loss.
- **Learning rate:** Controls step size; too small means slow convergence; too large means divergence.

- **Practical considerations:** Gradient descent is simple, scalable, and works well for large datasets.
- **Modern variants:** SGD, momentum, Adam, and other variants improve convergence in practice.

Appendix: Mathematical Notation

Symbol	**Definition**
θ	Parameter vector (what we're optimizing)
$J(\theta)$	Cost/loss function
$\nabla J(\theta)$	Gradient (vector of partial derivatives)
α	Learning rate (step size)
n	Number of data points
m	Number of parameters/features
X	Design matrix ($n \times m$)
y	Target vector ($n \times 1$)
H	Hessian matrix (matrix of second derivatives)
λ	Eigenvalue