Module 5: Optimization and Gradient Descent

Machine Learning Course

Contents

1 Introduction to Optimization in Machine Learning

1.1 The Optimization Framework

In machine learning, we typically choose a model parameterized by w by minimizing a loss function $L(w)$ that depends on the training data. This optimization problem is central to most machine learning algorithms.

1.2 Common Loss Functions

Different machine learning tasks use different loss functions:

- **Linear Regression:**
$$L(w) = \sum_{i=1}^{n} \left(y^{(i)} - w \cdot x^{(i)}\right)^2$$

- **Logistic Regression:**
$$L(w) = \sum_{i=1}^{n} \ln \left(1 + e^{-y^{(i)}(w \cdot x^{(i)})}\right)$$
where $y^{(i)} \in \{-1, 1\}$.

1.3 Local Search Methods

The default approach to solving these minimization problems is through local search:

1. Initialize w arbitrarily
2. Repeat until convergence:
   (a) Find some w' close to w with $L(w') < L(w)$
   (b) Move w to w'

Local search methods work particularly well when the loss function is convex, which we'll explore in detail later in this module.

2 Convexity

2.1 Definition and Intuition

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if for all $a, b \in \mathbb{R}^d$ and $0 < \theta < 1$:

$$f(\theta a + (1-\theta)b) \leq \theta f(a) + (1-\theta) f(b)$$

Intuitively, this means that the line segment connecting any two points on the graph of the function lies above or on the graph.

A function is strictly convex if strict inequality holds for all $a \neq b$. Conversely, f is concave if $-f$ is convex.

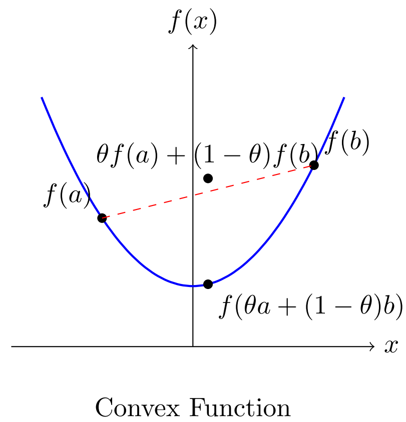

Figure 1: Illustration of convexity: The function value at any point on the line segment connecting two points is less than or equal to the weighted average of the function values at those points.

2.2 Why Convexity Matters in Optimization

Convexity ensures that any local minimum is a global minimum. This property is crucial for the success of local search and gradient-based optimization methods.

2.3 Checking Convexity

One Variable: Second Derivative Test

A twice-differentiable function $f : \mathbb{R} \to \mathbb{R}$ is convex if and only if $f''(x) \geq 0$ for all x in the domain.

Multivariate: Hessian and PSD Matrices

For $f : \mathbb{R}^d \to \mathbb{R}$, the Hessian matrix $H(x)$ is defined as:

$$H_{jk}(x) = \frac{\partial^2 f}{\partial x_j \partial x_k}(x)$$

A twice-differentiable function is convex if and only if its Hessian is positive semidefinite (PSD) everywhere.

2.4 Positive Semidefinite (PSD) Matrices

A symmetric matrix M is PSD if $x^T M x \geq 0$ for all $x \in \mathbb{R}^d$.

Properties:

- Diagonal matrix is PSD if all diagonal entries ≥ 0.
- If M is PSD and $c > 0$, then cM is PSD.
- If M, N are PSD, then $M + N$ is PSD.

- M is PSD iff $M = UU^T$ for some U.
- All covariance matrices are PSD.

2.5 Worked Examples

Example 1: Convexity of $f(x) = \|x\|^2$

$$f(x) = \|x\|^2 = \sum_{i=1}^{d} x_i^2$$

$$\frac{\partial f}{\partial x_j} = 2x_j$$

$$\frac{\partial^2 f}{\partial x_j \partial x_k} = 2\delta_{jk}$$

So the Hessian is $2I$, which is positive definite. Therefore, $f(x)$ is strictly convex.

Example 2: Convexity of $f(z) = (u \cdot z)^2$

$$f(z) = (u \cdot z)^2$$

$$\frac{\partial f}{\partial z_j} = 2(u \cdot z)u_j$$

$$\frac{\partial^2 f}{\partial z_j \partial z_k} = 2u_j u_k$$

So the Hessian is $2uu^T$, which is PSD since for any x, $x^T(2uu^T)x = 2(u^T x)^2 \geq 0$.

Example 3: Is $M = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ PSD? Let $x = (x_1, x_2)^T$:

$$x^T M x = (x_1 + x_2)^2 \geq 0$$

So M is PSD.

Example 4: Is $M = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$ PSD? Eigenvalues are 3 and -1. Since one eigenvalue is negative, M is not PSD.

Example 5: Convexity of Least Squares Loss

$$L(w) = \sum_{i=1}^{n} (y^{(i)} - w \cdot x^{(i)})^2$$

The Hessian is $2\sum_{i=1}^{n} x^{(i)}(x^{(i)})^T$, a sum of PSD matrices, so L is convex.

Example 6: Convexity of Logistic Regression Loss

$$L(w) = \sum_{i=1}^{n} \ln(1 + e^{-y^{(i)}(w \cdot x^{(i)})})$$

Each term $\ell(z) = \ln(1 + e^{-z})$ has $\ell''(z) > 0$, so L is convex.

3 Multivariate Differentiation

For $f : \mathbb{R}^d \to \mathbb{R}$:

- **Gradient:** $\nabla f(w) = \left(\frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_d}\right)^T$
- **Hessian:** $H_{jk}(w) = \frac{\partial^2 f}{\partial w_j \partial w_k}$

Example 1: $F(w) = 3w_1w_2 + w_3$ **for** $w \in \mathbb{R}^3$

$$\frac{\partial F}{\partial w_1} = 3w_2, \quad \frac{\partial F}{\partial w_2} = 3w_1, \quad \frac{\partial F}{\partial w_3} = 1$$

So,

$$\nabla F(w) = (3w_2, 3w_1, 1)^T$$

Example 2: $F(w) = w \cdot x$ **where** x **is fixed**

$$\frac{\partial F}{\partial w_j} = x_j$$

So,

$$\nabla F(w) = x$$

Example 3: Second Derivative Matrix of $F(w) = \|w\|^2$

$$F(w) = \sum_{j=1}^{d} w_j^2$$

$$\frac{\partial^2 F}{\partial w_j \partial w_k} = 2\delta_{jk}$$

So the Hessian is $2I$.

4 Gradient Descent

4.1 The Gradient and Its Geometric Meaning

The gradient points in the direction of steepest increase of a function. Moving in the negative gradient direction decreases the function most rapidly.

4.2 Gradient Descent Algorithm

1. Initialize $w_0 = 0$, $t = 0$
2. While $\|\nabla L(w_t)\| > \epsilon$ (not converged):
   (a) $w_{t+1} = w_t - \eta_t \nabla L(w_t)$
   (b) $t = t + 1$

Here, $\eta_t > 0$ is the step size (learning rate).

4.3 Rationale: Local Linearity and Descent Direction

For small u,

$$L(w + u) \approx L(w) + u \cdot \nabla L(w)$$

Choosing $u = -\eta \nabla L(w)$ for small η ensures $L(w + u) < L(w)$.

4.4 How to Set Step Size η_t?

- **Constant step size:** $\eta_t = \eta$
- **Diminishing step size:** $\eta_t = \frac{\eta_0}{1+\beta t}$ or $\eta_t = \frac{\eta_0}{\sqrt{t}}$
- **Backtracking line search:** Iteratively decrease η_t until $L(w_t - \eta_t \nabla L(w_t)) < L(w_t)$
- **Adaptive methods:** Adjust η_t based on gradient history (e.g., AdaGrad, Adam)

4.5 Gradient Descent for Logistic Regression: Full Derivation

Given $(x^{(i)}, y^{(i)}) \in \mathbb{R}^d \times \{-1, 1\}$,

$$L(w) = \sum_{i=1}^{n} \ln(1 + e^{-y^{(i)}(w \cdot x^{(i)})})$$

$$
\begin{aligned}
\nabla L(w) &= \nabla \sum_{i=1}^{n} \ln(1 + e^{-y^{(i)}(w\cdot x^{(i)})}) && (1)\\
&= \sum_{i=1}^{n} \nabla \ln(1 + e^{-y^{(i)}(w\cdot x^{(i)})}) && (2)\\
&= \sum_{i=1}^{n} \frac{1}{1 + e^{-y^{(i)}(w\cdot x^{(i)})}} \cdot \nabla e^{-y^{(i)}(w\cdot x^{(i)})} && (3)\\
&= \sum_{i=1}^{n} \frac{1}{1 + e^{-y^{(i)}(w\cdot x^{(i)})}} \cdot e^{-y^{(i)}(w\cdot x^{(i)})} \cdot (-y^{(i)}x^{(i)}) && (4)\\
&= -\sum_{i=1}^{n} \frac{e^{-y^{(i)}(w\cdot x^{(i)})}}{1 + e^{-y^{(i)}(w\cdot x^{(i)})}} \cdot y^{(i)}x^{(i)} && (5)\\
&= -\sum_{i=1}^{n} \frac{1}{1 + e^{y^{(i)}(w\cdot x^{(i)})}} \cdot y^{(i)}x^{(i)} && (6)\\
&= -\sum_{i=1}^{n} P(Y \neq y^{(i)}|x^{(i)}, w) \cdot y^{(i)}x^{(i)} && (7)
\end{aligned}
$$

Update rule:

$$w_{t+1} = w_t - \eta_t \nabla L(w_t)$$

5 Variants of Gradient Descent

5.1 Decomposable Loss Functions

Many loss functions decompose as $L(w) = \sum_{i=1}^{n} \ell(w; x^{(i)}, y^{(i)})$.

5.2 Stochastic Gradient Descent (SGD)

Update using a single data point:

$$w_{t+1} = w_t - \eta_t \nabla \ell(w_t; x^{(i)}, y^{(i)})$$

5.3 Mini-Batch Gradient Descent

Update using a batch B:

$$w_{t+1} = w_t - \eta_t \sum_{(x,y)\in B} \nabla \ell(w_t; x, y)$$

5.4 Comparison Table: GD, SGD, Mini-Batch GD

6 Convergence Properties

6.1 Convergence Guarantees for Convex Functions

For convex $L(w)$ and appropriate step sizes, gradient descent converges to the global minimum.

Method	Update Rule	Advantages	Disadvantage
Gradient Descent	$w_{t+1} = w_t - \eta_t \nabla L(w_t)$	Stable, accurate	Slow for large d
SGD	$w_{t+1} = w_t - \eta_t \nabla \ell(w_t; x^{(i)}, y^{(i)})$	Fast, low memory, escapes local minima	Noisy, may not
Mini-Batch GD	$w_{t+1} = w_t - \eta_t \sum_{(x,y)\in B} \nabla \ell(w_t; x, y)$	Balanced, parallelizable	Needs batch siz

Table 1: Comparison of gradient descent variants

6.2 Rates for GD, SGD, and Accelerated Methods

- GD: $O(1/T)$ for general convex, linear for strongly convex.
- SGD: $O(1/\sqrt{T})$ in expectation.
- Accelerated (e.g., Nesterov): $O(1/T^2)$ for smooth convex.

6.3 Practical Convergence Criteria (Stopping Conditions)

- $\|\nabla L(w_t)\| < \epsilon$
- $|L(w_{t+1}) - L(w_t)| < \delta$
- Maximum number of iterations

7 Advanced Optimization Methods

7.1 Momentum

$$v_{t+1} = \gamma v_t + \eta \nabla L(w_t)$$

$$w_{t+1} = w_t - v_{t+1}$$

where $\gamma \in [0, 1)$.

7.2 Adaptive Learning Rates

- AdaGrad: Per-parameter learning rates, decreases for frequent features.
- RMSProp: Exponential moving average of squared gradients.
- Adam: Combines momentum and RMSProp.

7.3 Second-Order Methods

- Newton's Method: $w_{t+1} = w_t - [H(w_t)]^{-1} \nabla L(w_t)$
- Quasi-Newton (e.g., BFGS, L-BFGS): Approximate inverse Hessian.

8 Practical Considerations

8.1 Initialization Strategies

- For convex problems, any w_0 will converge.
- For non-convex (e.g., neural nets), initialization affects which minimum is found.
- Common: random, Xavier/Glorot, He initialization.

8.2 Regularization and Its Effect

Adding regularization (e.g., L2: $\lambda\|w\|^2$) can improve convexity and generalization.

8.3 Non-Convex Optimization: Challenges and Context

Many modern ML models (e.g., deep neural networks) are non-convex. While gradient descent can still work well in practice, there are no guarantees of finding the global minimum.

8.4 Tips for Debugging and Tuning Optimization

- Monitor loss and gradients.
- Try different learning rates and batch sizes.
- Use validation data to check for overfitting.
- Visualize convergence when possible.

9 Practical Considerations

9.1 Initialization Strategies

- For convex problems, any w_0 will converge.
- For non-convex (e.g., neural nets), initialization affects which minimum is found.
- Common: random, Xavier/Glorot, He initialization.

9.2 Regularization and Its Effect

Adding regularization (e.g., L2: $\lambda\|w\|^2$) can improve convexity and generalization.

9.3 Non-Convex Optimization: Challenges and Context

Many modern ML models (e.g., deep neural networks) are non-convex. While gradient descent can still work well in practice, there are no guarantees of finding the global minimum.

9.4 Tips for Debugging and Tuning Optimization

- Monitor loss and gradients.
- Try different learning rates and batch sizes.
- Use validation data to check for overfitting.
- Visualize convergence when possible.