Optimization for Machine Learning

Reinforcement Learning (INF8250AE)
Fall 2025

Polytechnique Montréal

Mathematical Formulation of Optimization

Optimization Problem:

$$\theta^* = \arg \min_{\theta \in \mathbb{R}^m} f(\theta)$$

or equivalently (for maximization):

$$\theta^* = \arg \max_{\theta \in \mathbb{R}^m} f(\theta)$$

- $f : \mathbb{R}^m \to \mathbb{R}$ is the objective (loss or reward) function.
- $\theta \in \mathbb{R}^m$ are the parameters to be optimized.
- Goal: find θ^* that minimizes (or maximizes) $f(\theta)$.
- Gradient descent/ascent and its variants are iterative methods to solve this.

Statistical Learning Formulation in RL

Setup:

- Parameters: $\theta \in \mathbb{R}^m$ (policy parameters or value function parameters).
- State: $s \in \mathcal{S}$, Action: $a \in \mathcal{A}$.
- Policy: $\pi_\theta(a \mid s)$, the probability of taking action a in state s.
- Reward function: $r(s, a)$ and/or return $R = \sum_t \gamma^t r_t$.

Objective: Maximize expected return

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\Big[R(\tau)\Big]$$

where τ denotes a trajectory $(s_0, a_0, r_0, s_1, \dots)$.

Policy Gradient: Gradient ascent on expected return

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, R_t\right]$$

- Analogous to maximum likelihood: we adjust θ to increase probability of "good" trajectories.

First-Order Optimality Condition

At an optimum:

$$\frac{\partial f(\theta^*)}{\partial \theta} = 0$$

- This means that the slope of the function vanishes at θ^*.
- It is a necessary condition for local minima, maxima, or saddle points.

Gradient:

$$\nabla_\theta f(\theta) = \left(\frac{\partial f}{\partial \theta_1}, \frac{\partial f}{\partial \theta_2}, \ldots, \frac{\partial f}{\partial \theta_m} \right)$$

- The gradient is a vector of all partial derivatives.
- It points in the direction of steepest increase of $f(\theta)$.
- Gradient descent moves in the opposite direction to reach a local minimum.

Gradient Descent Algorithm

Goal: Find parameters θ that minimize $f(\theta)$.

Algorithm (for $i = 1, 2, \dots$):

1. Initialize parameters $\theta^{(0)}$ (random or heuristic).
2. Compute gradient:
$$g_i = \nabla_\theta f(\theta^{(i)})$$
3. Update rule:
$$\delta_i \leftarrow -\eta\, g_i$$
$$\theta^{(i+1)} \leftarrow \theta^{(i)} + \delta_i$$
4. Repeat until convergence (or stopping criterion).

Notes:

- η = learning rate (step size).
- Too large $\eta \rightarrow$ divergence; too small $\eta \rightarrow$ slow progress.
- Stopping criteria: small gradient, max iterations, or loss tolerance.

Why Gradient Descent Works (via Taylor expansion)

First/Second-Order Taylor at x:

$$f(x+\Delta) \;\approx\; f(x) + \nabla f(x)^\top \Delta \;+\; \tfrac{1}{2}\Delta^\top \nabla^2 f(\xi)\,\Delta$$

Descent step: choose $\Delta = -\eta \nabla f(x)$.

$$f(x-\eta\nabla f(x)) \;\approx\; f(x) - \eta\|\nabla f(x)\|^2 + \tfrac{\eta^2}{2}\,\nabla f(x)^\top \nabla^2 f(\xi)\,\nabla f(x)$$

Smoothness bound (Descent Lemma): if ∇f is L-Lipschitz,

$$f(x+\Delta) \le f(x) + \nabla f(x)^\top \Delta + \tfrac{L}{2}\|\Delta\|^2.$$

Plugging $\Delta = -\eta\nabla f(x)$:

$$f(x-\eta\nabla f(x)) \;\le\; f(x) - \eta\big(1 - \tfrac{L\eta}{2}\big)\,\|\nabla f(x)\|^2.$$

Conclusion: For $0 < \eta < \frac{2}{L}$, we have

$$f(x_{k+1}) \le f(x_k) - c\,\|\nabla f(x_k)\|^2 \quad (c = \eta(1 - \tfrac{L\eta}{2}) > 0),$$

so each step *decreases* f unless $\nabla f(x_k) = 0$.
Gradient ascent: apply the same argument to $-f$ to get an increase guarantee.

Momentum Method (Polyak, Heavy-ball)

Introduce a velocity term δ_i:

$$\delta_i = -\eta \nabla_\theta \mathcal{L}(\theta_{i-1}) + \alpha \delta_{i-1}$$

$$\theta_i = \theta_{i-1} + \delta_i$$

- $\alpha \in [0, 1)$ is the momentum coefficient.
- Reuses part of the previous update.
- Accelerates learning in consistent directions.

Intuition

- Imagine rolling a ball down a hill.
- Gradient descent: step-by-step, always reacts to slope.
- Momentum: keeps velocity, smooths oscillations.

Key Benefit

Momentum speeds up convergence and stabilizes training.

Variants

Nesterov Accelerated Gradient (NAG):

$$\delta_i = -\eta \nabla_\theta \mathcal{L}(\theta_{i-1} + \alpha \delta_{i-1}) + \alpha \delta_{i-1}$$

$$\theta_i = \theta_{i-1} + \delta_i$$

- Looks ahead before computing gradient.
- More accurate update direction.

Pros and Cons of Optimization Methods

Vanilla GD

- Pros:
  - Simple
  - Intuitive
- Cons:
  - Slow
  - Oscillates in narrow valleys

Momentum

- Pros:
  - Faster convergence
  - Smooth updates
  - Exploits consistent gradients
- Cons:
  - Sensitive to α
  - Can overshoot minima

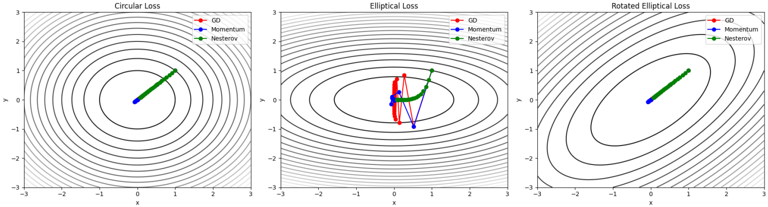
NAG

- Pros:
  - Looks ahead
  - More stable
  - Often better minima
- Cons:
  - Slightly more complex
  - Extra gradient computation

Trajectory of Different Optimizers

Concept: Different optimization algorithms follow different paths when minimizing a loss function.

- **Gradient Descent (GD):** May oscillate, especially in elongated valleys.
- **Momentum:** Smooths oscillations, faster convergence along consistent directions.
- **Nesterov:** Looks ahead, can overshoot if learning rate or momentum is large.

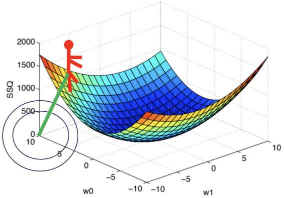


Regularization

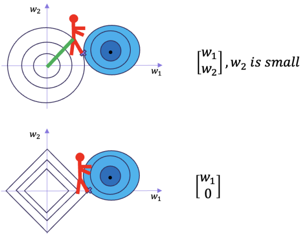
Concept: Regularization helps prevent overfitting by penalizing overly complex models. **Regularized Optimization Problem:**

$$\theta^* = \arg\min_{\theta} \left[\mathcal{L}(\theta) + \lambda R(\theta) \right]$$

- Without regularization: model fits noise → poor generalization
- With regularization: smoother model → better generalization



$H(W) = w_1 x + w_2 x^2$



Gradient Descent Variants

1. **Batch Gradient Descent**
   - Computes gradient using **all training samples**:

   $$\theta \leftarrow \theta - \eta \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \mathcal{L}_i(\theta)$$

   - Pros: stable, accurate gradient
   - Cons: slow for large datasets

2. **Stochastic Gradient Descent (SGD)**
   - Computes gradient using **one random sample** at a time:

   $$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_i(\theta)$$

   - Pros: fast, can escape shallow local minima
   - Cons: noisy updates, may oscillate

3. **Mini-batch Gradient Descent:** compromise between batch and stochastic — uses small subsets of data.

Important Function Derivatives in RL/ML

Sigmoid:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\frac{d\sigma(x)}{dx} = \frac{d}{dx}\frac{1}{1+e^{-x}}$$

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2}$$

$$\frac{d\sigma(x)}{dx} = \sigma(x)(1-\sigma(x))$$

Softmax:

$$\text{Softmax}(\vec{x})_i = s_i = \frac{e^{x_i}}{\sum_{k=1}^{d} e^{x_k}}$$

$$\frac{\partial s_i}{\partial x_i} = s_i(1-s_i),$$

$$\frac{\partial s_i}{\partial x_j} = -s_i s_j,\ i \neq j$$

$$\frac{\partial s_i}{\partial x_j} = \begin{cases} s_i(1-s_i), & i = j \\ -s_i s_j, & i \neq j \end{cases}$$

Function of Linear Operation:

$$\vec{h} = f(W\vec{x} + \vec{b}),$$

$$\vec{z} = W\vec{x} + \vec{b}$$

$$z_i = \sum_{j=1}^{d} W_{ij} x_j + b_i$$

$$\frac{\partial \vec{h}}{\partial W_{ij}} = \frac{\partial f}{\partial z_i} \cdot x_j$$