MLOpt, CSCI 4140/6960 – Spring 2026

Lecture 1: Introduction

Alex Gittens

Course website:
cs.rpi.edu/~gittea/old-site/teaching/spring2026/mlandopt.html

Outline

Course Overview

Course Logistics

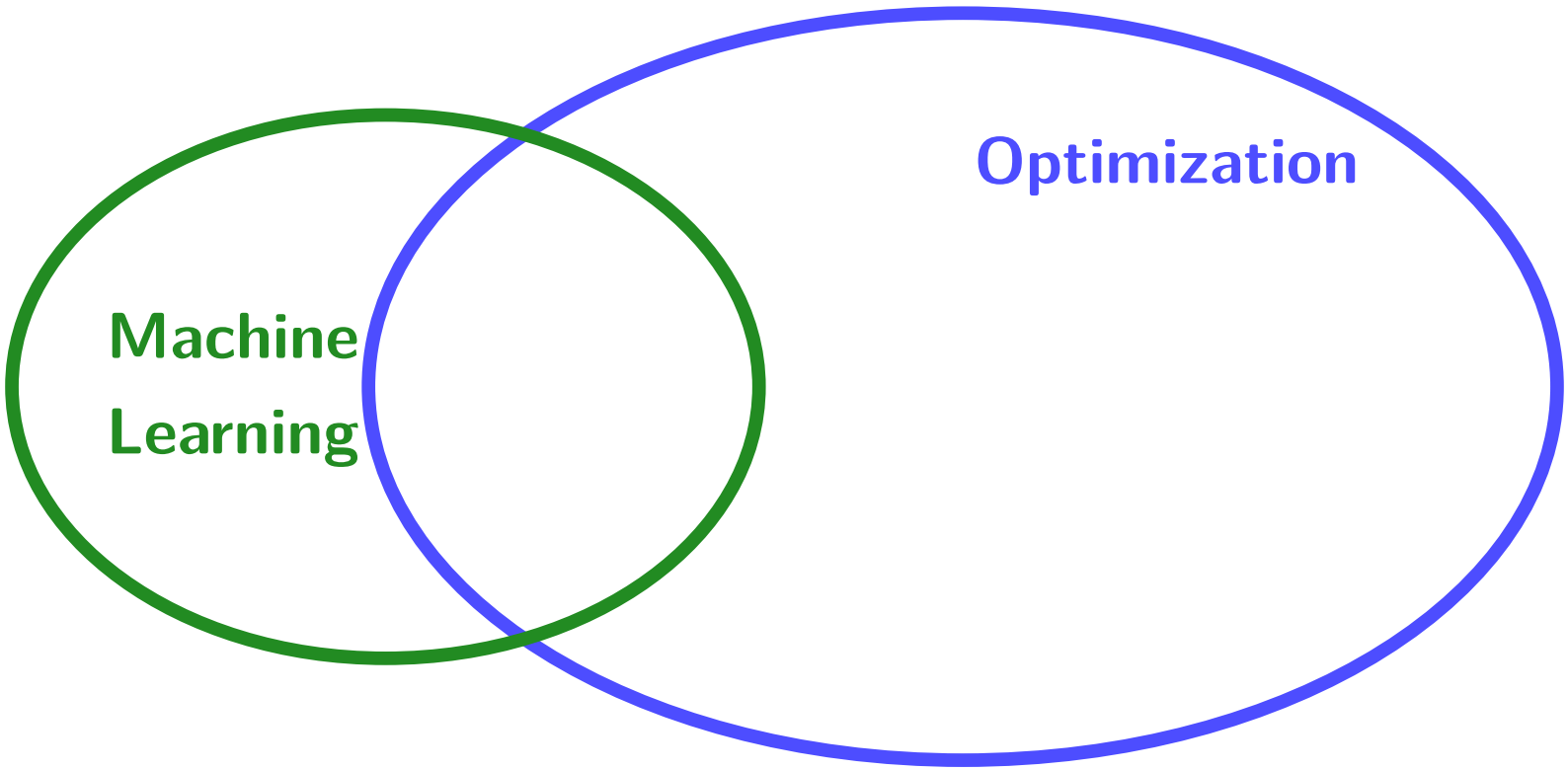
What is Optimization?

From ML to Optimization

Solving Optimization Problems

What is This Course About?

The Big Idea: Modern machine learning = solving optimization problems



Course Arc:

1. Optimization Foundations
2. Convex Optimization Algorithms
3. Neural Networks & Deep Learning

Learning Outcomes

By the end of this course, you will:

- **Formulate** ML tasks as optimization problems
- **Implement** gradient descent and second-order solvers
- **Understand** convergence guarantees and trade-offs
- **Build** modern deep learning architectures (CNNs, RNNs, Transformers)
- **Apply** these techniques to real-world problems

Prerequisites: Linear algebra, algorithms, basic probability at the level of MFML.

See {Mathematics of Machine Learning} for a refresher.

Course Structure

Website: `cs.rpi.edu/~gittea/.../mlandopt.html`

Lectures: Tue/Fri 12pm-2pm ET, JEC 5119

Instructors:

- Prof. Gittens: Tue 2pm-3pm ET, Fri 11am-12pm (Lally 316)
- Huzaifa Arif (TA): arifh@rpi.edu

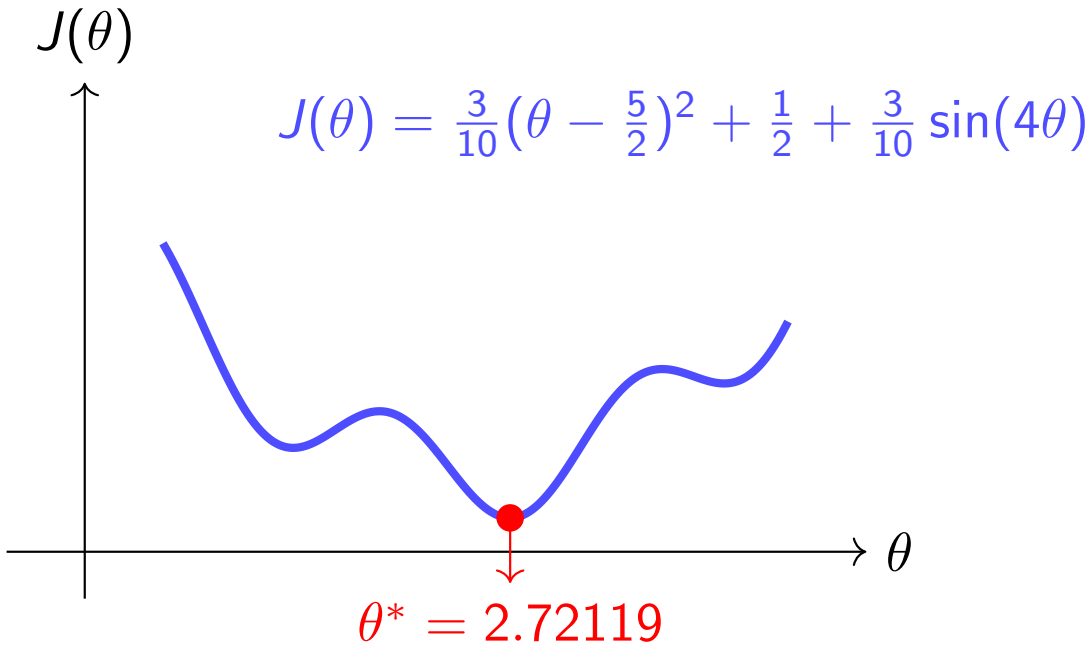
Assessment:

- 6 Homeworks, Midterm, Final, Project

Tools: Python, PyTorch, NumPy, standard ML libraries

Optimization: The Core Problem

Optimization: Finding extrema or extremal values of some objective



Notation:

$$\theta^* \in \arg\min_{\theta \in \Theta} J(\theta) \quad \text{or} \quad \theta^* \in \arg\max_{\theta \in \Theta} J(\theta)$$

The Machine Learning Recipe

General Framework: Empirical Risk Minimization (ERM)

Given training data: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$

Choose:

- A model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ parameterized by $\boldsymbol{\theta}$
- A loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ measuring prediction error

Optimization Problem

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i)$$

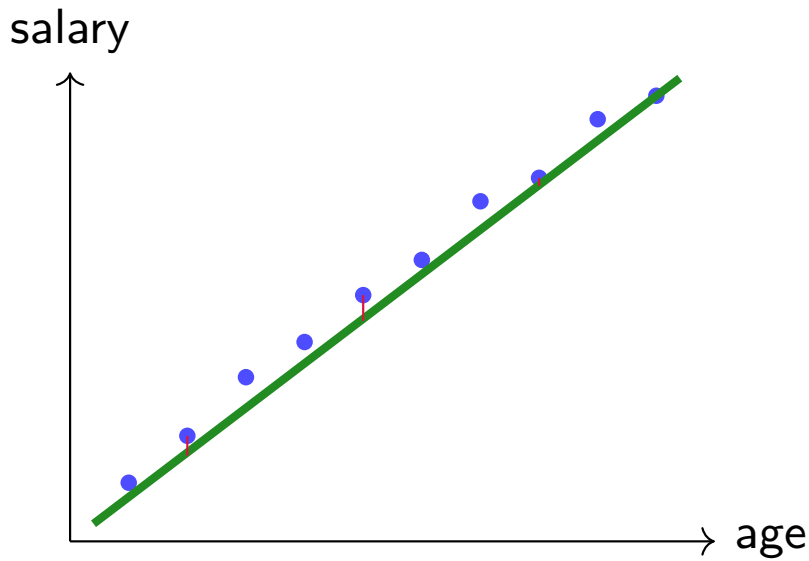
Goal: Find parameters that minimize average loss on training data

Example 1: Linear Regression

Problem: Predict continuous values (e.g., salary from age)

Setup:

- Data: $\{(\text{age}_i, \text{salary}_i)\}_{i=1}^n$
- Model: $f_x(\text{age}) = x \cdot \text{age}$ where $x \in \mathbb{R}$
- Loss: Squared error $\ell(\hat{y}, y) = (\hat{y} - y)^2$



Objective: $J(x) = \frac{1}{2} \sum_{i=1}^n (\text{salary}_i - x \cdot \text{age}_i)^2$

Linear Regression: Vector Formulation

With multiple predictors $\mathbf{z}_1, \ldots, \mathbf{z}_d$ (age, education, latitude, ...):

Stack data into matrices:

$$\underbrace{\begin{bmatrix} \text{salary}_1 \\ \text{salary}_2 \\ \vdots \\ \text{salary}_n \end{bmatrix}}_{\mathbf{b} \in \mathbb{R}^n} = \underbrace{\begin{bmatrix} \text{age}_1 & \text{edu}_1 & \cdots & \text{lat}_1 \\ \text{age}_2 & \text{edu}_2 & \cdots & \text{lat}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \text{age}_n & \text{edu}_n & \cdots & \text{lat}_n \end{bmatrix}}_{A \in \mathbb{R}^{n \times d}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}}_{\mathbf{x} \in \mathbb{R}^d}$$

Optimization Problem:

$$\mathbf{x}^* \in \arg\min_{\mathbf{x} \in \mathbb{R}^d} \|A\mathbf{x} - \mathbf{b}\|_2^2$$

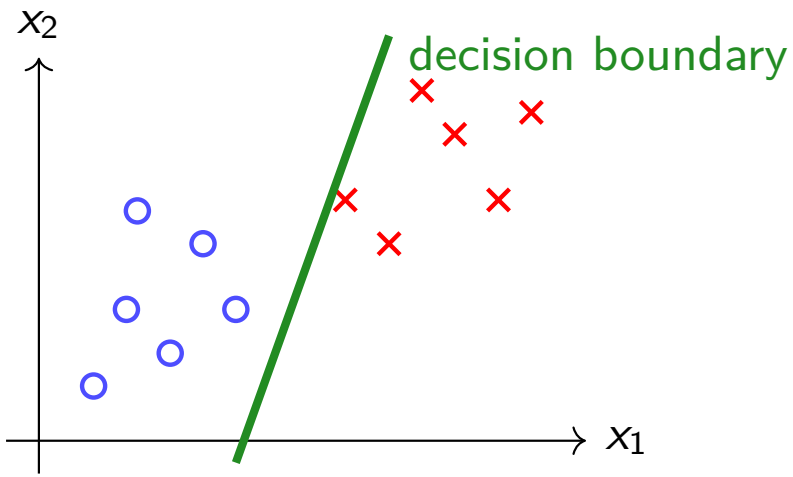
where $\|\mathbf{z}\|_2^2 = \sum_{i=1}^n z_i^2$ is the **Euclidean norm squared**

Example 2: Logistic Regression (Classification)

Problem: Predict binary outcomes (spam/not spam, disease/healthy)

Setup:

- Data: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $y_i \in \{-1, +1\}$
- Model: $P(y = 1|\mathbf{x}) = \sigma(\boldsymbol{\theta}^T \mathbf{x})$ where $\sigma(z) = \frac{1}{1+e^{-z}}$
- Loss: Logistic loss $\ell(\mathbf{x}, y) = \log(1 + e^{-y\boldsymbol{\theta}^T \mathbf{x}})$



Objective: $J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n \log(1 + e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i}) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2$

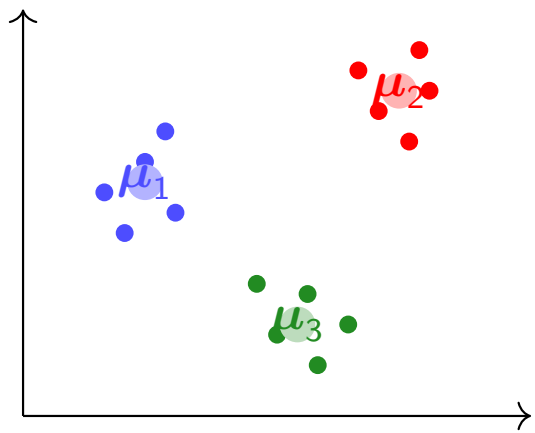
Key difference: No closed-form solution! Need iterative algorithms.

Example 3: K-Means Clustering (Unsupervised)

Problem: Group similar data points (customer segmentation, image compression)

Setup:

- Data: $\{\mathbf{x}_i\}_{i=1}^n$ (unlabeled!)
- Goal: Find K cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and assignments $C_1, \ldots, C_K$



Objective: $J(\{\boldsymbol{\mu}_k\}, \{C_k\}) = \sum_{k=1}^K \sum_{i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2$

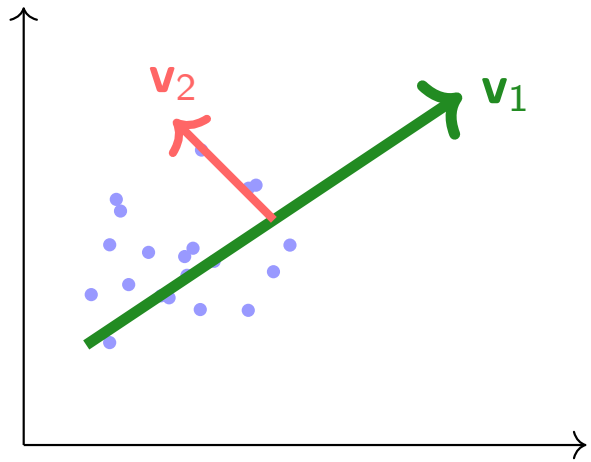
Challenge: Nonconvex! Algorithms find local minima (often good enough).

Example 4: Principal Component Analysis

Problem: Find low-dimensional representation of high-dimensional data

Setup:

- Data: $\{\mathbf{x}_i\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^d$
- Goal: Find direction $\mathbf{v} \in \mathbb{R}^d$ capturing maximum variance



Optimization Problem:

$$\mathbf{v}^* \in \arg\max_{\|\mathbf{v}\|_2=1} \mathbf{v}^T \left(\sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T \right) \mathbf{v}$$

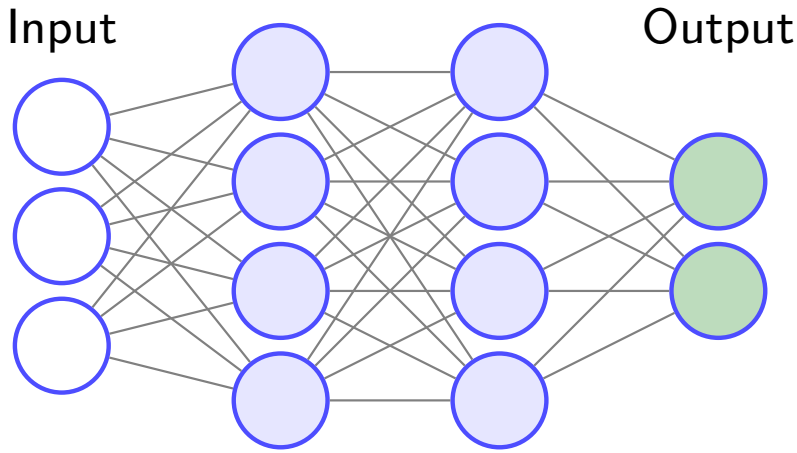
Solution: Leading eigenvector (constrained optimization, nonconvex)

Example 5: Neural Networks (Coming Attraction)

Problem: Learn complex nonlinear patterns (image recognition, language)

Setup:

- Model: $f_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma_L(W_L \sigma_{L-1}(\cdots \sigma_1(W_1 \mathbf{x}) \cdots))$
- Parameters: $\boldsymbol{\theta} = (W_1, \ldots, W_L)$ (millions to billions of parameters!)



Objective: $J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i)$

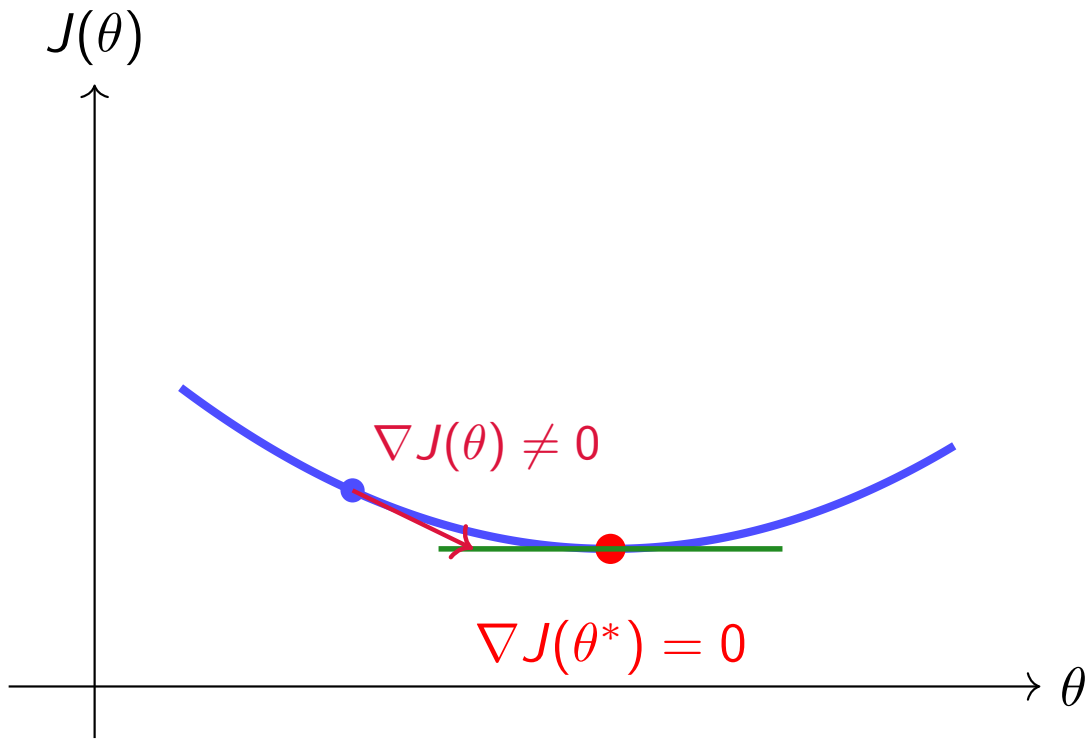
Challenges: Nonconvex, high-dimensional, requires stochastic optimization (second half of course!)

How Do We Solve These Problems?

First-order condition: Minimizers satisfy $\nabla J(\boldsymbol{\theta}) = \mathbf{0}$

where the gradient is

$$[\nabla J(\boldsymbol{\theta})]_i = \frac{\partial J}{\partial \theta_i}(\boldsymbol{\theta})$$



Challenge: For most ML problems, $\nabla J(\boldsymbol{\theta}) = \mathbf{0}$ is a nonlinear system—no closed form!

Closed-Form Solution: Least Squares

For linear regression $J(\mathbf{x}) = \|A\mathbf{x} - \mathbf{b}\|_2^2$:

Compute gradient:

$$\nabla J(\mathbf{x}) = 2A^T(A\mathbf{x} - \mathbf{b})$$

Set to zero:

$$\nabla J(\mathbf{x}) = \mathbf{0} \quad \Longleftrightarrow \quad A^T A\mathbf{x} = A^T\mathbf{b}$$

These are the **normal equations**.

Solution

If $A^T A$ is invertible:

$$\mathbf{x}^* = (A^T A)^{-1} A^T \mathbf{b}$$

If $A^T A$ is not invertible (e.g., more features than data points):

$$\mathbf{x}^* = (A^T A)^{\dagger} A^T \mathbf{b} \quad \text{where } (A^T A)^{\dagger} \text{ is the pseudoinverse}$$

Iterative Solution: Gradient Descent

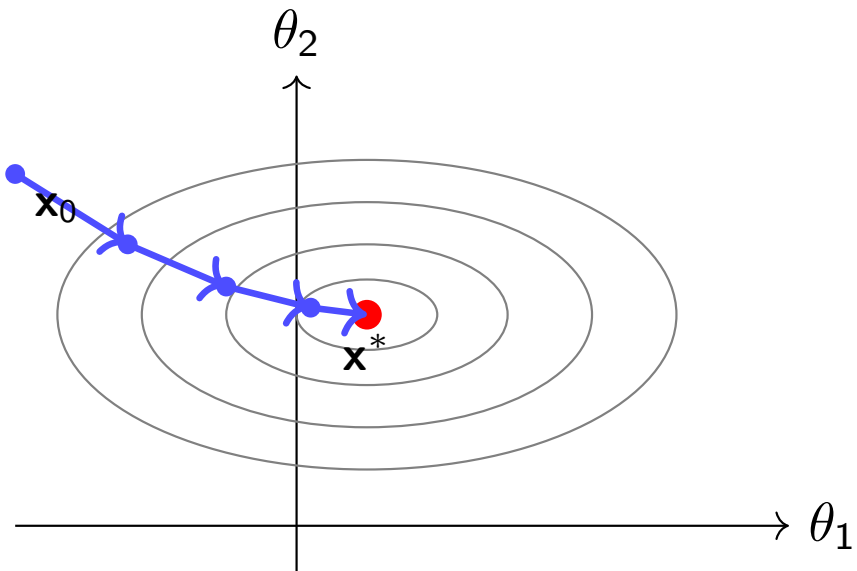
Idea: If $\nabla J(\mathbf{x}) = \mathbf{0}$ has no closed form, use iteration

Gradient Descent Algorithm

Input: Initial point $\mathbf{x}_0$, step size $\mu > 0$, iterations T
For $t = 0, 1, \ldots, T-1$:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \mu \nabla J(\mathbf{x}_t)$$

Intuition: Move in direction of steepest descent



Convergence Guarantees

Question: When does gradient descent converge to $\mathbf{x}^*$?

Answer (simplified): Under suitable assumptions on J, μ, and T:

- If J is **convex** and **smooth**: $J(\mathbf{x}_T) - J(\mathbf{x}^*) \leq O(1/T)$
- If J is **strongly convex** and smooth: $\|\mathbf{x}_T - \mathbf{x}^*\| \leq O(e^{-cT})$

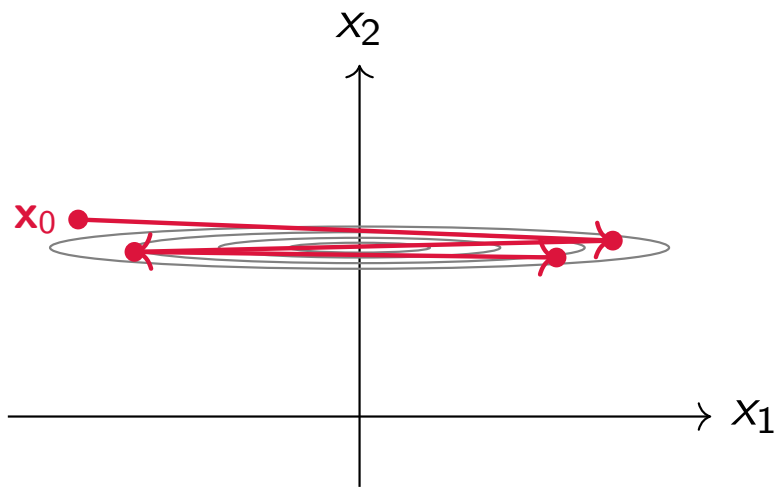
Coming up in this course:

- What do convex/smooth/strongly convex mean?
- How to choose step size μ?
- What about stochastic/noisy gradients?
- What about nonconvex problems (neural networks)?

Poor Behavior Example

Not all problems are equally easy! Consider:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^2} \frac{1}{2}\|D\mathbf{x}\|_2^2 \quad \text{where} \quad D = \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix}$$



Problem: Different curvature in different directions $\Rightarrow$ oscillation!

Solution: Second-order methods (Newton, quasi-Newton), adaptive gradient descent

Questions?

Next time: Modeling decisions, hypothesis spaces, losses, regularization