Introduction to Optimization

Cong Ma

University of Chicago, Winter 2026

An optimization problem

We write an optimization problem in the form

$$\min_{x \in \mathcal{C}} \; f(x),$$

where

- $x \in \mathbb{R}^n$ are the *decision variables*,
- $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*,
- $\mathcal{C} \subseteq \mathbb{R}^n$ is the *constraint/feasible set*.

Optimal solution: $x^\star \in \mathcal{C}$ such that

$$f(x^\star) \leq f(x) \quad \forall x \in \mathcal{C}.$$

Examples

- Least-squares and its regularized variants
- Matrix completion
- Empirical risk minimization and deep learning

Least-squares regression

Given data $(a_i, b_i) \in \mathbb{R}^n \times \mathbb{R}$, $i = 1, \ldots, k$, solve

$$\min_{x \in \mathbb{R}^n} \ \frac{1}{2k} \sum_{i=1}^{k} (a_i^\top x - b_i)^2$$

Equivalently,

$$\min_{x \in \mathbb{R}^n} \ \frac{1}{2k} \|Ax - b\|_2^2,$$

where

$$A = \begin{bmatrix} a_1^\top \\ \vdots \\ a_k^\top \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix}.$$

Interpretation: fit a linear model by minimizing squared prediction error.

Least-squares: geometric view

- Ax lies in the column space of A
- We are projecting b onto $\mathrm{col}(A)$

$$x^\star = \arg\min_x \|Ax - b\|_2 \quad \Longleftrightarrow \quad Ax^\star = \Pi_{\mathrm{col}(A)} b$$

Key property: smooth, convex objective with a unique minimizer (when $A^\top A$ is invertible).

Solving least-squares

Optimality condition:

$$A^\top (Ax - b) = 0 \quad \Longleftrightarrow \quad A^\top A x = A^\top b$$

Closed-form solution (when invertible):

$$x^\star = (A^\top A)^{-1} A^\top b$$

- Reliable and efficient numerical algorithms exist
- Computation depends on problem size and structure
- This is a *mature* optimization problem

Why regularize?

Least squares can behave poorly when:

- features are highly correlated
- n is large relative to k
- data is noisy

Symptoms:

- large coefficients
- unstable predictions

Ridge regression

$$\min_{x \in \mathbb{R}^n} \ \frac{1}{2k}\|Ax - b\|_2^2 + \lambda \|x\|_2^2$$

- $\lambda > 0$ controls the strength of regularization
- Penalizes large coefficients

Equivalent view:

$$\min_x \|Ax - b\|_2^2 \quad \text{s.t.} \quad \|x\|_2^2 \leq t$$

Properties of ridge regression

- Objective is smooth and strongly convex
- Unique solution always exists
- Closed-form solution:

$$x^\star = (A^\top A + 2k\lambda I)^{-1} A^\top b$$

- Improves numerical stability

Tradeoff: bias vs variance.

Lasso

$$\min_{x \in \mathbb{R}^n} \ \frac{1}{2k}\|Ax - b\|_2^2 + \lambda \|x\|_1$$

- $\|x\|_1 = \sum_{j=1}^{n} |x_j|$
- Encourages sparse solutions

Why does Lasso promote sparsity?

- ℓ_1 penalty has *corners*
- Optimal solutions often occur at corners
- Corners correspond to zero coordinates

Consequence: automatic variable selection.

Optimization perspective

- Objective is convex but **non-smooth**
- No closed-form solution
- Requires iterative algorithms

Key lesson: changing the regularizer changes both

- statistical behavior
- optimization difficulty

Motivation: The Netflix Challenge

- Netflix wants to recommend movies to users
- Users rate only a small fraction of movies
- Goal: *predict missing ratings accurately*
- This became the famous **Netflix Prize** problem

Key question: How do we infer a large number of missing entries from very few observations?

Data as a Matrix

- Let $M \in \mathbb{R}^{n \times m}$
  - Rows = users
  - Columns = movies
  - M_{ij} = rating user i gives movie j
- We observe only entries in a set Ω

$$\Omega \subset \{1, \ldots, n\} \times \{1, \ldots, m\}$$

Task: Predict M_{ij} for $(i, j) \notin \Omega$

Why Is This Hard?

- The matrix is huge (millions of users, thousands of movies)
- Most entries are missing (over 99%)
- Naively fitting all entries is impossible

Key insight: User preferences are *structured*

Low-Rank Structure

- Users can be described by a few latent factors
  - e.g. action vs romance, comedy vs drama
- Movies are also described by the same factors

$$M \approx UV^\top$$

- $U \in \mathbb{R}^{n \times r}$ (user factors)
- $V \in \mathbb{R}^{m \times r}$ (movie factors)
- $r \ll \min(n, m)$

Conclusion: M is approximately **low rank**

Matrix Completion Problem

We only observe M_{ij} for $(i,j) \in \Omega$.

Ideal formulation:

$$\min_{X} \ \operatorname{rank}(X) \quad \text{s.t.} \quad X_{ij} = M_{ij}, \ (i,j) \in \Omega$$

- Matches observed ratings exactly
- Chooses the simplest (lowest-rank) explanation

Why Rank Minimization Is Hard

- Rank is:
  - Non-convex
  - Discontinuous
  - NP-hard to optimize
- Not suitable for large-scale problems

Optimization lesson: We often replace hard objectives with *tractable surrogates*

Convex Relaxation: Nuclear Norm

- Replace rank with the **nuclear norm**

$$\|X\|_* = \sum_k \sigma_k(X)$$

- $\sigma_k(X) =$ singular values
- Convex envelope of rank on bounded sets

Convex formulation:

$$\min_X \ \|X\|_* \quad \text{s.t.} \quad X_{ij} = M_{ij},\ (i,j) \in \Omega$$

Noisy Observations

In practice:

- Ratings are noisy
- Exact fitting is undesirable

Regularized formulation:

$$\min_X \frac{1}{2} \sum_{(i,j)\in\Omega} (X_{ij} - M_{ij})^2 + \lambda \|X\|_*$$

- Data fitting term
- Complexity control via nuclear norm

Alternative: Factorized Formulation

Instead of optimizing over X directly:

$$X = UV^\top$$

$$\min_{U,V} \sum_{(i,j)\in\Omega} (u_i^\top v_j - M_{ij})^2 + \lambda(\|U\|_F^2 + \|V\|_F^2)$$

- Non-convex
- Much more scalable
- Used in practice by Netflix teams

Clustering example: communities in networks

- Data: a graph of interactions (friendships, emails, citations)
- Goal: partition nodes into **clusters/communities**
- Application: social networks, biology, recommender systems

Input: adjacency matrix $A \in \{0,1\}^{n \times n}$, where $A_{ij} = 1$ if there is an edge.

Stochastic Block Model (SBM)

- Each node has a hidden label $z_i \in \{1, 2\}$ (community)
- Edges are generated independently given labels:

$$\mathbb{P}(A_{ij} = 1 \mid z_i = z_j) = p, \qquad \mathbb{P}(A_{ij} = 1 \mid z_i \neq z_j) = q$$

with $p > q$.

Inference goal: recover labels $(z_1, \ldots, z_n)$ from A.

Optimization view: maximum likelihood

For balanced two-way clustering, let $x \in \{\pm 1\}^n$ encode labels ($x_i = +1$ vs -1), and enforce balance $\mathbf{1}^\top x = 0$.

A common surrogate objective (related to likelihood / cut):

$$\max_{x \in \{\pm 1\}^n,\ \mathbf{1}^\top x = 0} \quad x^\top A x$$

- **Discrete, nonconvex** optimization
- Direct search is impossible when n is large

Relaxation idea: from discrete to continuous

Relax the constraint $x \in \{\pm 1\}^n$ to a sphere constraint:

$$\max_{\|x\|_2^2 = n,\ \mathbf{1}^\top x = 0} \quad x^\top A x$$

This becomes an eigenvector problem:

- solution uses the **top eigenvector** of (centered) A
- then **round** by $\operatorname{sign}(x_i)$

Optimization lesson: discrete $\to$ relaxation $\to$ rounding.

Empirical Risk Minimization (ERM)

Given data $(x_1, y_1), \ldots, (x_n, y_n)$, ERM solves

$$\min_{\theta \in \Theta} \quad \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i)$$

- $\theta =$ model parameters
- $f_\theta =$ prediction function
- $\ell =$ loss function

Unifying view: least squares, logistic regression, SVMs, neural networks.

ERM includes familiar problems

- **Least squares:**

$$\ell(f_\theta(x), y) = \tfrac{1}{2}(f_\theta(x) - y)^2, \quad f_\theta(x) = \theta^\top x$$

- **Logistic regression:**

$$\ell(f_\theta(x), y) = \log(1 + \exp(-y f_\theta(x)))$$

- **Regularization:**

$$\frac{1}{n}\sum_{i=1}^{n} \ell(\cdot) + \lambda \|\theta\|^2$$

Same optimization template, different modeling choices.

Deep learning = large-scale optimization

In deep learning:

- f_θ is a deep neural network
- θ contains millions (or billions) of parameters
- The loss is typically nonconvex

$$\min_\theta \quad \frac{1}{n} \sum_{i=1}^{n} \ell\big(f_\theta(x_i), y_i\big)$$

Key fact: training a neural network is an optimization problem.

Why optimization matters for deep learning

- Objective is **nonconvex**
- Data is **massive**
- Exact minimization is impossible

Yet simple methods like gradient descent work remarkably well.

Central questions:

- Why does optimization succeed?
- What structure are we exploiting?

A recurring pattern

- Start with a real-world problem
- Formulate an optimization problem
- Identify structure
- Choose algorithms accordingly

This pattern will repeat throughout the course.

Topics

- **Formulation:** convex optimization, nonconvex optimization
- **Algorithms:** gradient descent, Newton's method
- **Applications:** examples in data fitting, statistical estimation, geometric problems, etc
- **Duality Theory**