15. Review of convex optimization

- Convex sets and functions
- Convex programming models
- Network flow problems
- Least squares problems
- Regularization and tradeoffs
- Duality
Convex sets

A set $C \subseteq \mathbb{R}^n$ is **convex** if for all $x, y \in C$ and all $0 \leq \alpha \leq 1$, we have: $\alpha x + (1 - \alpha)y \in C$.

- every line segment must be contained in the set
- can include boundary or not
- can be finite or not
Examples

1. Polyhedron
   - A linear inequality $a_i^T x \leq b_i$ is a *halfspace*.
   - Intersections of halfspaces form a polyhedron: $Ax \leq b$.

Halfspace in 3D

Polyhedron in 3D.
Examples

2. Ellipsoid

- A quadratic form looks like: $x^T Q x$
- If $Q \succ 0$ (positive definite; all eigenvalues positive), then the set of $x$ satisfying $x^T Q x \leq b$ is an ellipsoid.
Examples

3. Second-order cone constraint

- The set of points satisfying $\|Ax + b\| \leq c^T x + d$ is called a second-order cone constraint.
- Example: robust linear programming

Second order cone: $\|x\| \leq y$

Constraints $a_i^T x + \rho \|x\| \leq b_i$
**Convex functions**

A function \( f : D \rightarrow \mathbb{R} \) is a **convex function** if:

1. the domain \( D \subseteq \mathbb{R}^n \) is a convex set
2. for all \( x, y \in D \) and \( 0 \leq \alpha \leq 1 \), the function \( f \) satisfies:
   \[
   f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y)
   \]

- any line segment joining points of \( f \) lies above \( f \).
- \( f \) is continuous, not necessarily smooth
- \( f \) is **concave** if \(-f\) is convex.
Convex programs

\[
\begin{align*}
\text{minimize} & \quad f_0(x) \\
\text{subject to:} & \quad f_i(x) \leq 0 \quad \text{for } i = 1, \ldots, m \\
& \quad h_j(x) = 0 \quad \text{for } j = 1, \ldots, r
\end{align*}
\]

- the domain is the set \( D \)
- the cost function is \( f_0 \)
- the inequality constraints are the \( f_i \) for \( i = 1, \ldots, m \).
- the equality constraints are the \( h_j \) for \( j = 1, \ldots, r \).
- **feasible set**: the \( x \in D \) satisfying all constraints.

A model is **convex** if \( D \) is a convex set, all the \( f_i \) are convex functions, and the \( h_j \) are affine functions (linear + constant).
Examples

1. Linear program (LP)
   - cost is affine
   - all constraints are affine
   - can be maximization or minimization

Important properties
- feasible set is a polyhedron
- can be optimal, infeasible, or unbounded
- optimal point occurs at a vertex


2. Convex quadratic program (QP)

- cost is a convex quadratic
- all constraints are affine
- must be a minimization

Important properties

- feasible set is a polyhedron
- optimal point occurs on boundary or in interior
Examples

3. Convex quadratically constrained QP (QCQP)

- cost is convex quadratic
- inequality constraints are convex quadratics
- equality constraints are affine

Important properties

- feasible set is an intersection of ellipsoids
- optimal point occurs on boundary or in interior
4. Second-order cone program (SOCP)

- cost is affine
- inequality constraints are second-order cone constraints
- equality constraints are affine

Important properties

- feasible set is convex
- optimal point occurs on boundary or in interior
Hierarchy of complexity

From simplest to most complicated:

1. linear program
2. convex quadratic program
3. convex quadratically constrained quadratic program
4. second-order cone program
5. semidefinite program
6. general convex program

Important notes

• more complicated just means that e.g. every LP is a SOCP (by setting appropriate variables to zero), but a general SOCP cannot be expressed as an LP.

• in general: strive for the simplest model possible
Network flow problems

- Each edge \((i, j) \in E\) has a flow \(x_{ij} \geq 0\).
- Each edge has a transportation cost \(c_{ij}\).
- Each node \(i \in \mathcal{N}\) is: a source if \(b_i > 0\), a sink if \(b_i < 0\), or a relay if \(b_i = 0\). The sum of flows entering \(i\) must equal \(b_i\).
- Find the flow that minimizes total transportation cost while satisfying demand at each node.
Network flow problems

- **Capacity constraints**: \( 0 \leq x_{ij} \leq q_{ij} \quad \forall (i, j) \in \mathcal{E}. \)
- **Balance constraint**: \( \sum_{j \in \mathcal{N}} x_{ij} = b_i \quad \forall i \in \mathcal{N}. \)
- **Minimize total cost**: \( \sum_{(i, j) \in \mathcal{E}} c_{ij} x_{ij} \)

We assume \( \sum_{i \in \mathcal{N}} b_i = 0 \) (balanced graph). Otherwise, add a dummy node with no cost to balance the graph.
Network flow problems

Expanded form:

\[
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \\
0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[A = \text{incidence matrix}\]
Integer solutions

minimize \[ c^T x \]
subject to: \[ Ax = b \]
\[ 0 \leq x \leq q \]

- If \( A \) is a totally unimodular matrix then if demands \( b_i \) and capacities \( q_{ij} \) are integers, the flows \( x_{ij} \) are integers.
- All incidence matrices are totally unimodular.
Examples

• **Transportation problem:** each node is a source or a sink

• **Assignment problem:** transportation problem where each source has supply 1 and each sink has demand 1.

• **Transshipment problem:** like a transportation problem, but it also has relay nodes (warehouses)

• **Shortest path problem:** single source, single sink, and the edge costs are the path lengths.

• **Max-flow problem:** single source, single sink. Add a feedback path with $-1$ cost and minimize the cost.
Least squares

- We want to solve $Ax = b$ where $A \in \mathbb{R}^{m \times n}$.

- Typical case of interest: $m > n$ (overdetermined). If there is no solution to $Ax = b$ we try instead to have $Ax \approx b$.

- The least-squares approach: make Euclidean norm $\|Ax - b\|$ as small as possible.

**Standard form:**

$$\min_{x} \|Ax - b\|^2$$

It’s an unconstrained convex QP.
Example: curve-fitting

- We are given noisy data points \((x_i, y_i)\).
- We suspect they are related by \(y = px^2 + qx + r\).
- Find the \(p, q, r\) that best agrees with the data.

Writing all the equations:

\[
\begin{align*}
y_1 & \approx px_1^2 + qx_1 + r \\
y_2 & \approx px_2^2 + qx_2 + r \\
     & \vdots \\
y_m & \approx px_m^2 + qx_m + r
\end{align*}
\]

\[
\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \approx \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ x_m^2 & x_m & 1 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}
\]

- Also called **regression**.
Regularization:

Additional penalty term added to the cost function to encourage a solution with desirable properties.

Regularized least squares:

\[
\minimize_x \| Ax - b \|^2 + \lambda R(x)
\]

- \( R(x) \) is the regularizer (penalty function)
- \( \lambda \) is the regularization parameter
- The model has different names depending on \( R(x) \).
Examples

\[
\minimize_x \| Ax - b \|^2 + \lambda R(x)
\]

1. If \( R(x) = \|x\|^2 = x_1^2 + x_2^2 + \cdots + x_n^2 \)
   It is called: \( L_2 \) regularization, Tikhonov regularization, or Ridge regression depending on the application. It has the effect of smoothing the solution.

2. If \( R(x) = \|x\|_1 = |x_1| + |x_2| + \cdots + |x_n| \)
   It is called: \( L_1 \) regularization or LASSO. It has the effect of sparsifying the solution (\( \hat{x} \) will have few nonzero entries).

3. \( R(x) = \|x\|_\infty = \max\{|x_1|, |x_2|, \ldots, |x_n|\} \)
   It is called \( L_\infty \) regularization and it has the effect of equalizing the solution (makes most components equal).
Suppose \( J_1 = \|Ax - b\|^2 \) and \( J_2 = \|Cx - d\|^2 \).

We would like to make both \( J_1 \) and \( J_2 \) small.

A sensible approach: solve the optimization problem:

\[
\min_x J_1 + \lambda J_2
\]

where \( \lambda > 0 \) is a (fixed) tradeoff parameter.

Then tune \( \lambda \) to explore possible results.

\[\begin{align*}
\text{When } \lambda &\to 0, \text{ we place more weight on } J_1 \\
\text{When } \lambda &\to \infty, \text{ we place more weight on } J_2
\end{align*}\]
• Pareto-optimal points can only improve in $J_1$ at the expense of $J_2$ or vice versa.
Example: Min-norm least squares

**Underdetermined case:** $A \in \mathbb{R}^{m \times n}$ is a wide matrix ($m \leq n$), so $Ax = b$ has infinitely many solutions.

- Look to make both $\|Ax - b\|^2$ and $\|x\|^2$ small

\[
\minimize_x \quad \|Ax - b\|^2 + \lambda \|x\|^2
\]

- In the limit $\lambda \to \infty$, we get $x = 0$

- In the limit $\lambda \to 0$, we get the min-norm solution:

\[
\minimize_x \quad \|x\|^2 \\
\text{subject to:} \quad Ax = b
\]
Duality

**Intuition:** Duality is all about finding solution bounds.

- If the primal problem is a minimization, all feasible points of the primal are *upper bounds* on the optimal solution.
- The dual problem is a maximization. All feasible points of the dual are *lower bounds* on the optimal solution.
Example: LP duality

Primal problem (P)

\[
\begin{align*}
\text{maximize} & \quad c^T x \\
\text{subject to:} & \quad A x \leq b \\
\text{and} & \quad x \geq 0
\end{align*}
\]

Dual problem (D)

\[
\begin{align*}
\text{minimize} & \quad b^T \lambda \\
\text{subject to:} & \quad A^T \lambda \geq c \\
\text{and} & \quad \lambda \geq 0
\end{align*}
\]

If \( x \) and \( \lambda \) are feasible points of (P) and (D) respectively:

\[
c^T x \leq p^* \leq d^* \leq b^T \lambda
\]

- In the case of LPs, the dual of the dual is the primal
Strong duality

We have **strong duality** if $p^* = d^*$

- When dealing with LPs, if either the primal or dual has a finite solution, then strong duality holds.

- When dealing with general convex programs, if there is a strictly feasible point then strong duality holds. This is called **Slater’s condition**.

These sorts of conditions that can guarantee strong duality are called **constraint qualifications**.
If the constraint \( f_i(x) \leq 0 \) has associated dual variable \( \lambda_i \), then \( f_i(x^*)\lambda_i^* = 0 \). This means that:

- If \( f_i(x^*) < 0 \) (loose constraint), then \( \lambda_i^* = 0 \)
- If \( \lambda_i^* > 0 \) (positive dual variable), then \( f_i(x^*) = 0 \)

**Sensitivity:** The size of \( \lambda_i \) indicates how much a change in the constraint \( f_i \) will affect the optimal cost.