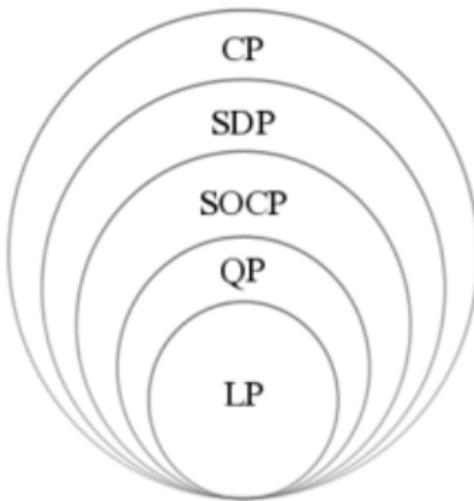


Optimization in Machine Learning

Constrained Optimization Linear Programming



Learning goals

- Definition and different forms of an LP
- Geometric intuition of LPs
- Characteristics of vertices
- Simplex algorithm



LINEAR PROGRAMMING

- **Linear program (LP):** optimization problem with **linear** objective function + **linear** constraints
- General form

$$\min_{\mathbf{x}} \mathbf{c}^T \mathbf{x} \quad \text{s.t. } \mathbf{A}_1 \mathbf{x} \leq \mathbf{b}_1, \mathbf{A}_2 \mathbf{x} \geq \mathbf{b}_2, \mathbf{A}_3 \mathbf{x} = \mathbf{b}_3$$

- Examples

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_1 \Leftrightarrow$$

$$\min_{\mathbf{x}, \mathbf{s}} \mathbf{1}^T \mathbf{s} \quad \text{s.t. } \pm (\mathbf{Ax} - \mathbf{b}) \leq \mathbf{s}$$

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_\infty \Leftrightarrow$$

$$\min_{\mathbf{x}, t} t \quad \text{s.t. } \pm (\mathbf{Ax} - \mathbf{b}) \leq t \mathbf{1}$$



LINEAR PROGRAMMING: STANDARD FORM

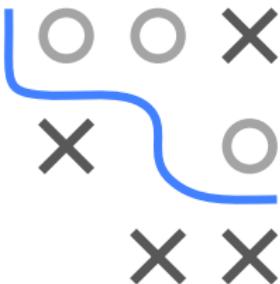
Standard Form

$$\min \mathbf{c}^T \mathbf{x} \quad \text{s.t. } \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$$



- $\mathbf{Ax} \geq \mathbf{b} \Leftrightarrow -\mathbf{Ax} \leq -\mathbf{b}$
- $\mathbf{Ax} = \mathbf{b} \Leftrightarrow \mathbf{Ax} \leq \mathbf{b}, -\mathbf{Ax} \leq -\mathbf{b}$
- $\mathbf{x} = \mathbf{x}^+ - \mathbf{x}^-$, where $\mathbf{x}^+ \geq \mathbf{0}, \mathbf{x}^- \geq \mathbf{0}$
- $\min_{\mathbf{x}} \mathbf{c}^T \mathbf{x} \Leftrightarrow \min_{\mathbf{x}^+, \mathbf{x}^-} [\mathbf{c}^T \quad -\mathbf{c}^T] \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \end{bmatrix}$

LINEAR PROGRAMMING: EQUALITY FORM



- Equality Form

$$\min \mathbf{c}^T \mathbf{x} \quad \text{s.t. } \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq 0$$

- By introducing slack variables \mathbf{s} :

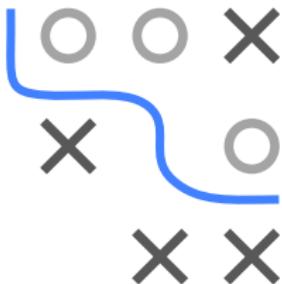
$$\mathbf{Ax} \leq \mathbf{b} \Leftrightarrow [\mathbf{A} \quad \mathbf{I}] \begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} = \mathbf{Ax} + \mathbf{s} = \mathbf{b}, \mathbf{s} \geq 0$$

$$\min_{\mathbf{x}} \mathbf{c}^T \mathbf{x} \Leftrightarrow \min_{\mathbf{x}, \mathbf{s}} [\mathbf{c}^T \quad \mathbf{0}^T] \begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix}$$

GEOMETRIC INTERPRETATION

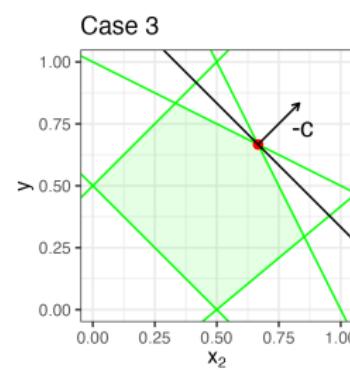
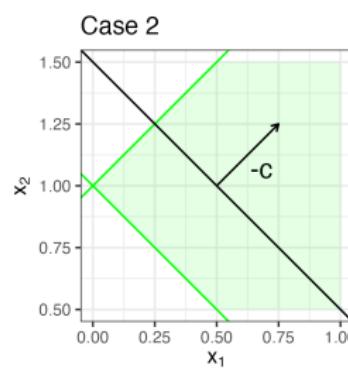
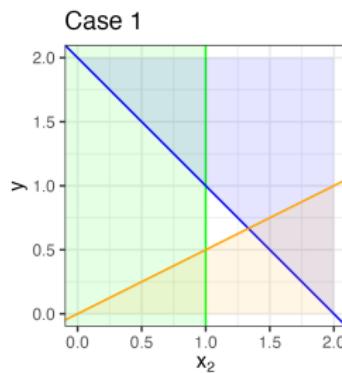
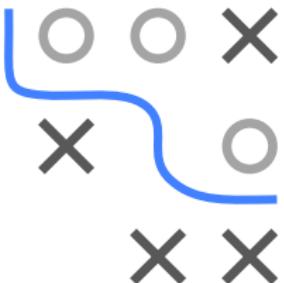
Feasible set

- Points $\{x : \mathbf{A}_i^T x = b_i\}$ form a hyperplane in \mathbb{R}^n
 \mathbf{A}_i is perpendicular to the hyperplane and called **normal vector**
- Points $\{x : \mathbf{A}_i^T x \leq b_i\}$ lie on one side of the hyperplane, which form a convex half-space
- Points satisfying **all** inequalities form a **convex polytope**
The intersection of convex sets is still convex
- Polytope $\{x : \mathbf{Ax} \leq \mathbf{b}, \mathbf{A} \in \mathbb{R}^{m \times n}\}$ is an **n -simplex**, i.e., convex hull of $n + 1$ **affinely independent** points, which we call vertices



GEOMETRIC INTERPRETATION

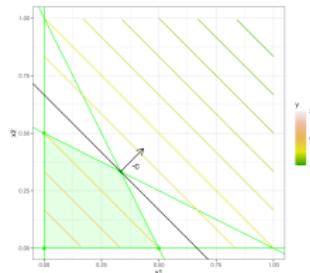
- There are 3 conditions for solving linear programming:
 - ① Feasible set is **empty** \Rightarrow LP is infeasible
 - ② Feasible set is **unbounded**
 - ③ Feasible set is **bounded** \Rightarrow LP has at least one solution



GEOMETRIC INTERPRETATION

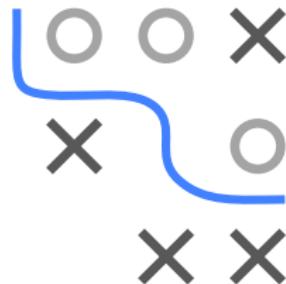
Case 3: LP is feasible and bounded

- Points on the interior: never optimal, can be improved by moving along $-\mathbf{c}$
- Points on faces/edges: can be optimal only if the face/edge is perpendicular to \mathbf{c}
- Points on faces/edges not perpendicular to \mathbf{c} : can be improved by moving along $-\mathbf{c}$
- Vertices: can also be optimal



VERTICES

- Assume rows of $\mathbf{A} \in \mathbb{R}^{m \times n}$ are linearly independent and $m \leq n$ to form a bounded non-empty feasible set
- $\mathbf{Ax} = \mathbf{b}$ imposes m equality constraints:
 - Each equality constraint reduces the dimension of the feasible set by 1
 - Starting with n -dim space, applying m independent equality constraints leaves a solution space of dim $n - m$
- $\mathbf{x} \geq \mathbf{0}$ imposes n non-negativity constraints



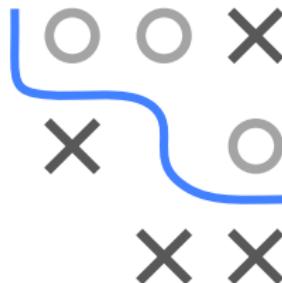
VERTICES

- While satisfying $\mathbf{Ax} = \mathbf{b}$, the indices of a vertex vector can be partitioned into two sets:
 - \mathcal{V} with $n - m$ elements: $i \in \mathcal{V} \Rightarrow x_i = 0$ (active constraints)
 - \mathcal{B} with m elements: $i \in \mathcal{B} \Rightarrow x_i \geq 0$
- We have $\mathbf{A}_{\mathcal{B}}^{m \times m} \mathbf{x}_{\mathcal{B}} = \mathbf{b} \Rightarrow \mathbf{x}_{\mathcal{B}} = \mathbf{A}_{\mathcal{B}}^{-1} \mathbf{b}$
- **Note:** While every vertex has an associated partition $(\mathcal{B}, \mathcal{V})$, not every partition corresponds to a vertex



SIMPLEX ALGORITHM

- The **simplex algorithm** solves LPs by moving from vertex to vertex of the feasible set, and produces an optimal vertex
- Operates on equality-form LPs $\mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$
Assume rows of $\mathbf{A} \in \mathbb{R}^{m \times n}$ are linearly independent and $m \leq n$
- Guaranteed to arrive at an optimal solution so long as the LP is feasible and bounded
- The simplex algorithm operates in two phases:
 - **Initialization** phase: identifies a vertex partition
 - **Optimization** phase: transitions between vertex partitions toward a partition corresponding to an optimal vertex



SIMPLEX ALGORITHM



- Construct a Lagrangian for the equality form:

$$L(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \mathbf{c}^T \mathbf{x} - \boldsymbol{\mu}^T \mathbf{x} - \boldsymbol{\lambda}^T (\mathbf{A}\mathbf{x} - \mathbf{b})$$

with $\boldsymbol{\mu} \geq \mathbf{0}$

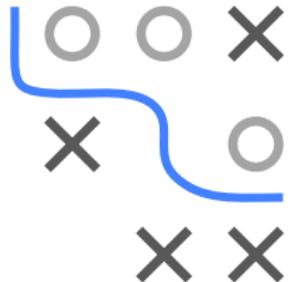
- Optimal solution satisfies $\frac{\partial L}{\partial \mathbf{x}} = 0$, i.e., $\mathbf{A}^T \boldsymbol{\lambda} + \boldsymbol{\mu} = \mathbf{c}$
- Decompose stationarity condition into \mathcal{B} and \mathcal{V} components:

$$\mathbf{A}^T \boldsymbol{\lambda} + \boldsymbol{\mu} = \mathbf{c} \quad \Rightarrow \quad \begin{cases} \mathbf{A}_{\mathcal{B}}^T \boldsymbol{\lambda} + \boldsymbol{\mu}_{\mathcal{B}} = \mathbf{c}_{\mathcal{B}} \\ \mathbf{A}_{\mathcal{V}}^T \boldsymbol{\lambda} + \boldsymbol{\mu}_{\mathcal{V}} = \mathbf{c}_{\mathcal{V}} \end{cases}$$

SIMPLEX ALGORITHM

- Choose $\mu_B = 0$ to satisfy $\mu \odot x = 0$, since for optimality, we need $\mu_i = 0$ when $x_i > 0$
- Compute λ from B :

$$\mathbf{A}_B^T \lambda + \underbrace{\mu_B}_{=0} = \mathbf{c}_B \implies \lambda = \mathbf{A}_B^{-T} \mathbf{c}_B$$



- Use this to obtain:

$$\mathbf{A}_V^T \lambda + \mu_V = \mathbf{c}_V$$

$$\mu_V = \mathbf{c}_V - \mathbf{A}_V^T \lambda$$

$$\mu_V = \mathbf{c}_V - (\mathbf{A}_B^{-1} \mathbf{A}_V)^T \mathbf{c}_B$$

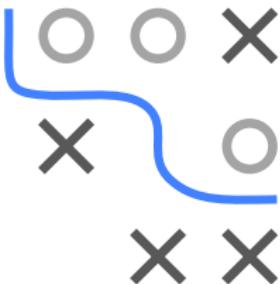
- Knowing μ_V allows us to assess optimality of the vertices
If μ_V contains negative components, then $\mu \geq \mathbf{0}$ is not satisfied and the vertex is suboptimal

SIMPLEX ALGORITHM: OPTIMIZATION PHASE

- Partition can be updated by swapping indices between \mathcal{B} and \mathcal{V}
Such a swap equates to moving from one vertex along an edge of the feasible set polytope to another vertex
- A transition $\mathbf{x} \rightarrow \mathbf{x}'$ between vertices must satisfy $\mathbf{Ax}' = \mathbf{b}$
- Starting with a partition defined by \mathcal{B} , choose an **entering index** $q \in \mathcal{V}$ that is to enter \mathcal{B} using one of the heuristics described below
- The new vertex \mathbf{x}' must satisfy:

$$\mathbf{Ax}' = \mathbf{A}_{\mathcal{B}} \mathbf{x}'_{\mathcal{B}} + \mathbf{A}_{\{q\}} \mathbf{x}'_q = \mathbf{A}_{\mathcal{B}} \mathbf{x}_{\mathcal{B}} = \mathbf{Ax} = \mathbf{b}$$

- One **leaving index** $p \in \mathcal{B}$ in $\mathbf{x}_{\mathcal{B}}$ becomes zero during the transition, and is replaced by the column of \mathbf{A} corresponding to index q
This action is referred to as **pivoting**



SIMPLEX ALGORITHM: MINIMUM RATIO TEST

- Solve for the new design point:

$$\mathbf{x}'_{\mathcal{B}} = \mathbf{x}_{\mathcal{B}} - \mathbf{A}_{\mathcal{B}}^{-1} \mathbf{A}_{\{q\}} x'_q$$

- A particular **leaving index** $p \in \mathcal{B}$ becomes active when:

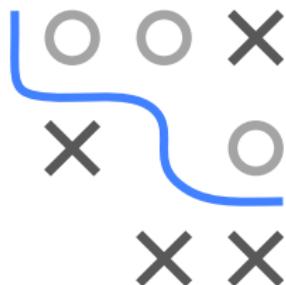
$$(\mathbf{x}'_{\mathcal{B}})_p = 0 = (\mathbf{x}_{\mathcal{B}})_p - (\mathbf{A}_{\mathcal{B}}^{-1} \mathbf{A}_{\{q\}})_p x'_q$$

and is thus obtained by increasing $x_q = 0$ to x'_q with:

$$x'_q = \frac{(\mathbf{x}_{\mathcal{B}})_p}{(\mathbf{A}_{\mathcal{B}}^{-1} \mathbf{A}_{\{q\}})_p}$$

- The leaving index is obtained using the **minimum ratio test**, which computes for each potential leaving index and selects the one with minimum x'_q

We then swap p and q between \mathcal{B} and \mathcal{V}



SIMPLEX ALGORITHM: EFFECT ON OBJECTIVE

- Effect of a transition on the obj. function can be computed using x'_q
- Objective function value at the new vertex:

$$\begin{aligned}\mathbf{c}^T \mathbf{x}' &= \mathbf{c}_B^T \mathbf{x}'_B + c_q x'_q \\ &= \mathbf{c}_B^T (\mathbf{x}_B - \mathbf{A}_B^{-1} \mathbf{A}_{\{q\}} x'_q) + c_q x'_q \\ &= \mathbf{c}_B^T \mathbf{x}_B - \mathbf{c}_B^T \mathbf{A}_B^{-1} \mathbf{A}_{\{q\}} x'_q + c_q x'_q \\ &= \mathbf{c}_B^T \mathbf{x}_B - (c_q - \mu_q) x'_q + c_q x'_q \\ &= \mathbf{c}^T \mathbf{x} + \mu_q x'_q\end{aligned}$$

using $\lambda = \mathbf{A}_B^{-T} \mathbf{c}_B$ and $\mathbf{A}_{\{q\}}^T \lambda = c_q - \mu_q$

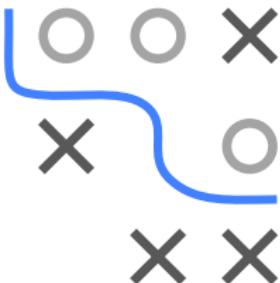


SIMPLEX ALGORITHM: OPTIMALITY CONDITION

- Choosing an entering index q decreases the obj. function value by

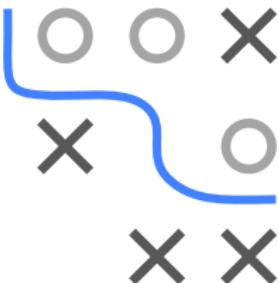
$$\mathbf{c}^T \mathbf{x}' - \mathbf{c}^T \mathbf{x} = \mu_q x'_q$$

- The objective function decreases only if μ_q is negative
- To progress toward optimality, we must choose an index $q \in \mathcal{V}$ such that μ_q is negative
- If all components of $\mu_{\mathcal{V}}$ are non-negative, we have found a global optimum



SIMPLEX ALGORITHM: HEURISTICS

- Since there can be multiple negative entries in μ_V , different heuristics can be used to select an entering index:
- **Greedy heuristic:** choose a q that maximally reduces $\mathbf{c}^T \mathbf{x}$
- **Dantzig's rule:** choose the q with the most negative entry in μ
 - Easy to calculate, but does not guarantee the maximum reduction in $\mathbf{c}^T \mathbf{x}$
 - Also sensitive to scaling of the constraints
- **Bland's rule:** choose the first q with a negative entry in μ
 - Tends to result in poor performance in practice when used alone
 - Helps prevent cycles (returning to a visited vertex without decreasing objective)
 - Usually used only after no improvements have been made for several iterations to break out of a cycle and ensure convergence



SIMPLEX ALGORITHM: EXAMPLE

$$\mathbf{A}_{\mathcal{V}} = \begin{pmatrix} 2 & 1 & 1 & 0 \\ -4 & 2 & 0 & 1 \end{pmatrix}, \mathbf{b} = (9, 2)^T, \mathbf{c} = (3, -1, 0, 0)^T$$

Solution:

$$\mathcal{V} = \{1, 2\}, \mathcal{B} = \{3, 4\}$$

$$\mathbf{x}_{\mathcal{B}} = \mathbf{A}_{\mathcal{B}}^{-1} \mathbf{b} = (9, 2)^T$$

$$\boldsymbol{\lambda} = \mathbf{A}_{\mathcal{B}}^{-1} \mathbf{c}_{\mathcal{B}} = \mathbf{0}$$

$$\boldsymbol{\mu}_{\mathcal{V}} = \mathbf{c}_{\mathcal{V}} - (\mathbf{A}_{\mathcal{B}}^{-1} \mathbf{A}_{\mathcal{V}})^T \mathbf{c}_{\mathcal{B}} = (3, -1)^T$$

$\boldsymbol{\mu}_{\mathcal{V}}$ contains negative elements, so our current \mathcal{B} is suboptimal.

We will pivot on the negative one with $q = 2, -\mathbf{A}_{\mathcal{B}}^{-1} \mathbf{A}_{\{q\}} = (1, 2)^T$.

This causes $x_4 = 0$, so updated $\mathcal{B} = \{2, 3\}$.

In the second iteration, we find $\mathbf{x}_{\mathcal{B}} = (1, 8)^T, \boldsymbol{\lambda} = (0, -\frac{1}{2})^T, \boldsymbol{\mu}_{\mathcal{V}} = (1, \frac{1}{2})^T$.

This is optimal with no negative entry, thus we have $x^* = (0, 1, 8, 0)^T$

