Lecture 4: Linear programming

Sondre Pedersen

January 17, 2025

Linear programming is a special case of constrained optimization. In some way it is the simplest, but still very easy. Lecturer quote: *"in two lectures, you will learn almost as much about linear programming as Indøk does in a full course."*

Plan:

- Brief recap: Linear algebra
- Linear programming (LP): Formulation and standard form
- KKT conditions for LP
- The dual LP, weak & strong duality

1 Linear algebra recap

Norms Vector norm: A mapping $||\cdot|| : \mathbb{R}^n \to \mathbb{R}^+$ that satisifes

- $||x|| = 0 \implies x = 0$
- $||x + z|| \leq ||x|| + ||z||$, for all $x, z \in \mathbb{R}^n$
- $||\alpha x|| = |\alpha| ||x||$, for all $\alpha \in \mathbb{R}$ and $x \in \mathbb{R}^n$

In words: measure the size of "something'.

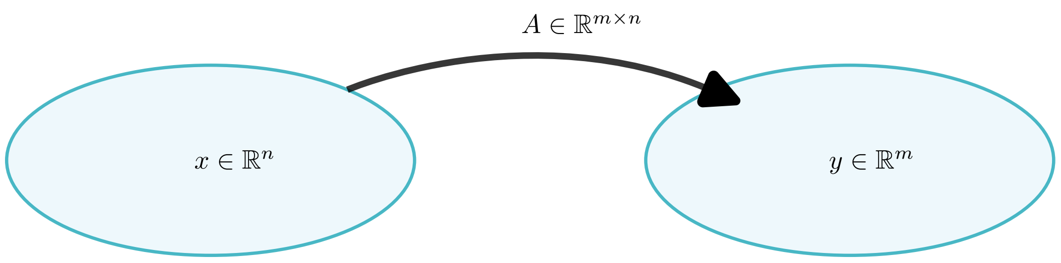
Common vector norms:

- 1-norm: $||x||_1 = |x_1| + |x_2| + \cdots + |x_2|$ (sum norm)
- 2-norm: $||x||_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$ (Euclidean norm)
- ∞-norm: $||x||_\infty = \max_{i=1,\dots,n} |x_i|$ (max norm)

Induced matrix norms, $A \in \mathbb{R}^{m\times n} : ||A||_p := \sup_{x\neq 0} \frac{||Ax||_p}{||x||_p}$:

- 1-norm: $||A||_1 = \max_{j=1,\dots,m} \sum_{i=1}^n |A_{ij}|$
- 2-norm: $||A||_2 = \lambda_{\max}(\sqrt{A^\top A})$
- ∞-norm: $||A||_1 = \max_{i=1,\dots,m} \sum_{j=1}^n |A_{ij}|$
- Frobenius-norm (not induced): $||A||_F = \sqrt{\sum_{i=1}^M \sum_{j=1}^N A_{ij}^2}$

Fundamental Theorem of Linear Algebra A matrix $A \in \mathbb{R}^{m\times n}$ is a mapping:



- Nullspace of A: $\text{Null}(A) = \{v \in \mathbb{R}^n | Av = 0\}$
- Columnspace of A: $\text{Range}(A) = \{w \in \mathbb{R}^m | w = Av\}$, for some $v \in \mathbb{R}^n$

Fundamental theorem of linear algebra: $\text{Null}(A) \oplus \text{Range}(A^\top) = \mathbb{R}^n$

Matrix Factorizations 'All' algorithms in this course involve solving linear equation systems:

$$Ax = b \qquad \implies \qquad x = A^{-1}b.$$

But in practive, **never** use the matrix inverse. It is inefficient and numerically unstable. Instead, use matrix factorizations: For a general matrix A: Use LU-decomposition (Gaussian elimination)

$$A = LU: \qquad Ax = LUx = b \qquad \implies \qquad Ly = b \qquad \implies \qquad Ux = y.$$

Due to triangular structure of L and U, we easily solve the two linear systems by substitution. Can think of this as Guassian elimination. With a positive definite matrix A, we should use Cholesky decomposition: $A = LL$. For symmetric, indefinite matrices use LDL-factorization instead.

Condition Number of a Matrix (when solving Ax = b)

Well-conditioned matrix: A small change in the input gives a small change in the output.

$$\begin{bmatrix}1 & 2\\ 1 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} = \begin{bmatrix}3\\ 2\end{bmatrix} \implies \begin{bmatrix}x_1\\ x_2\end{bmatrix} = \begin{bmatrix}1\\ 1\end{bmatrix}$$
$$\begin{bmatrix}1 & 2\\ 1 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} = \begin{bmatrix}3.00001\\ 2\end{bmatrix} \implies \begin{bmatrix}x_1\\ x_2\end{bmatrix} = \begin{bmatrix}0.99999\\ 1.00001\end{bmatrix}$$

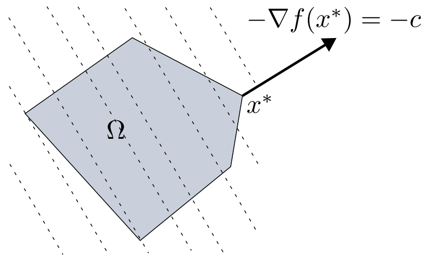
Ill-conditioned matrix: A small change in the input gives a large change in the output.

$$\begin{bmatrix}1.00001 & 1\\ 1 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} = \begin{bmatrix}2.00001\\ 2\end{bmatrix} \implies \begin{bmatrix}x_1\\ x_2\end{bmatrix} = \begin{bmatrix}1\\ 1\end{bmatrix}$$
$$\begin{bmatrix}1.00001 & 1\\ 1 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} = \begin{bmatrix}2\\ 2\end{bmatrix} \implies \begin{bmatrix}x_1\\ x_2\end{bmatrix} = \begin{bmatrix}0\\ 2\end{bmatrix}$$

To check if the matrix is well conditioned, do $\kappa(A) = ||A||||A^{-1}||$. A small κ (1-100) means that the matrix is well-conditioned. A large number (>10 000) means that it is ill-conditioned.

2 Linear Programming

$$\min c^\top x \qquad \text{subject to} \qquad \begin{cases} a_i^\top x - b_i = 0, \ i \in \mathcal{E} \\ a_i^\top x - b_i \geq 0, \ i \in \mathcal{I} \end{cases}.$$



Standard form LP $\min_{x\in\mathbb{R}^n} c^\top x \qquad$ subject to $\begin{cases} Ax = b \\ x \geq 0 \end{cases}$

Standard form is convenient for theory development. It is also great for the Simplex algorithm. Aditionally, it is good practice for learning about transformations. These can be important in general for modelling optimization. For example, you might be able to transform a non-convex problem into a convex one.

Trick 1: slack variables

$$\min_{x,z} c^\top x \qquad \text{s.t.} \qquad Ax + z = b, \qquad z \geq 0.$$

Trick 2: Split x into positive and negative part

$$x = x^+ - x^-, \ x^+ \geq 0, \ x^- \geq 0.$$

An example: $\begin{bmatrix}3\\-5\end{bmatrix} = \begin{bmatrix}3\\0\end{bmatrix} - \begin{bmatrix}0\\5\end{bmatrix}$. For the full problem, it now looks like

$$\min_{\begin{bmatrix}x^+ & x^- & z\end{bmatrix}^\top} \begin{bmatrix}c^\top & -c^\top & 0\end{bmatrix}\begin{bmatrix}x^+\\x^-\\z\end{bmatrix} \qquad \text{s.t.} \qquad \begin{cases}\begin{bmatrix}A & -A & I\end{bmatrix}\begin{bmatrix}x^+\\x^-\\z\end{bmatrix} = b\\ \begin{bmatrix}x^+\\x^-\\z\end{bmatrix} \geq 0\end{cases}.$$

We went from a problem with n variables to a problem with 2-3 times as many variables. But it turns out the algorithms are well suited to solving them still.

LPs can have no solution if either (i) the feasible set is empty, or (ii) if the feasible set is unbounded in the direction of improving objective function.

The feasible set is a polytope - a convex set with flat faces. There are three different cases for solutions. No solution, one solution (a corner of the polytope) or an edge.

3 KKT conditions for linear programming

The KKT conditions are checked slightly differently for linear programming. That is, introduce another type of lagrange multiplier for the $x \geq 0$ conditions, s

$$\mathcal{L}(x, \lambda, s) = c^\top x - \lambda^\top(Ax - b) - s^\top x.$$

KKT:

$$\begin{aligned}\nabla_x \mathcal{L}(x^*, \lambda^*, s^*) =& c - A^\top\lambda^* - s^* = 0\\ Ax^* &= b\\ x^* &\geq 0\\ s^* &\geq 0\\ s_i^* \times x_i^* &= 0,\ i = 1, \dots, n\end{aligned}$$

KKT for LPs is necessary and sufficient

Proof: Assume $\overline{x}$ feasible, x^*, λ^*, s^* fullfils KKT.

$$\begin{aligned}c^\top\overline{x} &= (A^\top\lambda^* + s^*)^\top\overline{\lambda}\\ &= {\lambda^*}^\top A\overline{x} + {s^*}^\top\overline{x}\\ &\geq {\lambda^*}^\top b\end{aligned}$$

$$\begin{aligned}c^\top x^* &= (A^\top\lambda^* + s^*)^\top x^*\\ &= {\lambda^*}^\top Ax^* + {s^*}^\top x^*\\ &= {\lambda^*}^\top b\end{aligned}$$

That is: $c^\top\overline{x} \geq c^\top x^*,\ \forall\overline{x} \in \Omega$. This means that x^* is a global solution.

4 The dual LP problem

Introduce the dual:

$$\max_{\lambda\in\mathbb{R}^m} b^\top\lambda \qquad \text{subject to} \qquad A^\top\lambda \leq c.$$

Rewrite for KKT: $\min_\lambda -b^\top\lambda$ s.t. $c - A^\top\lambda \geq 0$. Lagrangian:

$$\overline{\mathcal{L}}(\lambda, x) = -b^\top\lambda - x^\top(c - A^\top\lambda).$$

KKT:

$$\begin{aligned}\nabla_\lambda \overline{\mathcal{L}}(\lambda^*, x^*) &= -b + Ax^* = 0 \\ c - A^\top \lambda^* &\geq 0 \\ x^* &\geq 0 \\ \lambda_i^* - [c - A^\top x^*]_i &= 0\end{aligned}$$

Define $s^* = c - A^\top \lambda^*$ Put this back into the conditions above, and we find that the dual and the primal are the same.
Weak dualtiy: $c^\top \overline{x} \geq b^\top \overline{\lambda}$

This says that the objective of the primal is larger than the objective of the dual. Strong duality is that they are equal.