ECON 289: Lecture 1
Linear Programming
(or, How to Find the Saddle Point)

Instructor: Ben Brooks

Spring, 2023

Overview for the course

- Topics:
  - Linear programming
  - Epistemic game theory
  - Robust predictions
  - Bayesian mechanism design
  - Robust mechanism design
- Evaluation:
  - Problem sets in Python/Gurobi
  - Final project/presentation in which you formulate a new model and analyze it numerically, using tools from the class
- Gestalt:
  - The synergy between simulation and analysis, especially in the context of linear models
    (which come up all the time in game theory and mechanism design)

Ordered fields

- Fix an ordered field $\mathbb{F}$
  - A set endowed with addition/multiplication,
  - Identities 0 and 1
  - Additive and multiplicative inverses (except there is no multiplicative inverse for 0)
  - A linear order $>$ that is compatible with addition/multiplication in the usual way
- Think of $\mathbb{R}$ or $\mathbb{Q}$
- But also includes more exotic ordered fields, e.g., algebraic numbers, rational functions, hyperreals
- (Why do I care that we are working with fields rather than $\mathbb{R}$? To emphasize the conceptual point that all of our arguments are **algebraic**, rather than **topological**)

Vectors and matrices

- $\mathbb{F}^n$ is the set of n-vectors
- Think of them as column vectors, but for the most part it won't matter
- For $x \in \mathbb{F}^n$, we write $x \geq 0$ if all coordinates are non-negative
- If x and y are vectors of same length, then $xy = \sum_{i=1}^{n} x_i y_i$
- $\mathbb{F}^{m \times n}$ is the set of matrices with m rows and n columns
- Given $A \in \mathbb{F}^{m \times n}$, A_i is the ith row and A^i is the ith column

Matrix/vector operations

- Given $x \in \mathbb{F}^n$, $y \in \mathbb{F}^m$, and $A \in \mathbb{F}^{m \times n}$, we define
  - Ax is the vector of products of x with rows of A
  - yA is the vector of products of y with columns of A
- NB I am following Gale (1960) in generally ignoring whether a vector is "row" or "column" (but when we build block matrices, we will regard them as row vectors)
- Transpose of $A \in \mathbb{F}^{m \times n}$ is the matrix $A^\top$, with $A^\top_{ij} = A_{ji}$
- Identity matrix $I \in \mathbb{F}^{m \times m}$ has $I_{ii} = 1$ and $I_{ij} = 0$ for $j \neq i$

Linear Programs

- A **linear program** (LP) is an finite-dimensional optimization problem (maximization or minimization) with a linear objective and linear constraints
- Any linear program can be put in the "canonical" or "standard"[1] form

$$\max_{x \in \mathbb{F}^n} cx \text{ s.t. } x \geq 0, Ax \leq b \qquad \text{(P)}$$

- Parameterized by $(A, b, c) \in \mathbb{F}^{m \times n} \times \mathbb{F}^m \times \mathbb{F}^n$
- The **dimension** is $m \times n$

[1] In scare quotes because there are many different "standards"

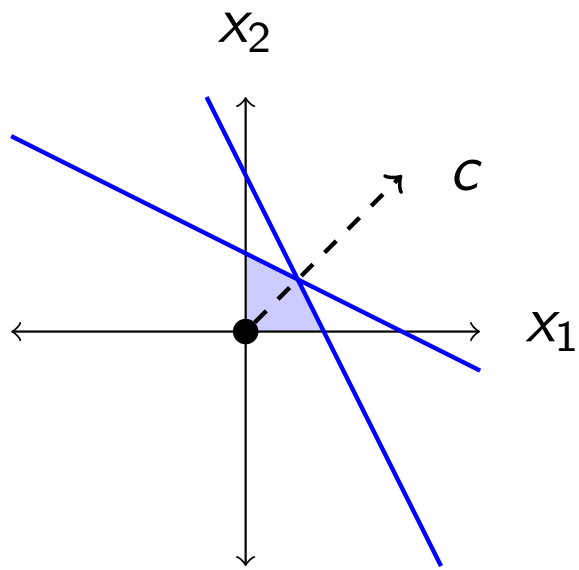
Manipulating LPs into standard form

- On your problem set, you will prove that any LP can be rewritten in standard form
- For example, any equality constraints of the form $Ax = b$ can be converted into a pair of inequality constraints $Ax \leq b$ and $-Ax \leq -b$
- NB In the other direction, $Ax \leq b$ can be rewritten as $Ax + z = b$, $z \geq 0$ (indeed, another "standard" form is to write the LP constraints as $Ax = b$)
- In some cases, optimization problems that don't appear to be LPs can be reformulated as such, e.g., an objective that is a ratio of linear functions, or constraints involving a max or min
- These kinds of programs can be rewritten as LPs by introducing **auxiliary variables**

Toy example

- Consider the LP

$$\max x_1 + x_2 \text{ s.t. } x_1 \geq 0,\ x_2 \geq 0$$
$$2x_1 + x_2 \leq 1, x_1 + 2x_2 \leq 1$$



- From the picture, it is self-evident what is the optimum: The northeast corner of the feasible set, at $x_1 = x_2 = 1/3$
- Optimal value is $x_1 + x_2 = 2/3$

More economic examples: Production

- From Dorfman, Samuelson, Solow (1958)
- The auto plant can produce cars and trucks
- The plant has four departments, each of which has 100% capacity; here are the percent capacity usages per vehicle

	Cars	Trucks
Metal stamping	0.004	0.00286
Engine Assembly	0.003	0.006
Car assembly	0.00444	0
Truck assembly	0	0.00667

- The net profit per car is \$300 and the net profit per truck is \$250
- How many cars and trucks should the plant produce, without exceeding plant capacity, to maximize profit?

Production formulated as an LP

- Cars x_1 and trucks x_2
- Constraints:

$$
\begin{aligned}
x_1 \geq 0,\ x_2 \geq 0;& \\
100 \geq 0.004x_1 + 0.00286x_2& \\
100 \geq 0.003x_1 + 0.006x_2& \\
100 \geq 0.00444x_1& \\
100 \geq 0.00667x_2&
\end{aligned}
$$

- Objective: Maximize $300x_1 + 250x_2$
- Solution: On your pset!

More economic examples: Transportation

- You own a firm that sells a certain divisible goods
- You have a collection of warehouses $i = 1, \ldots, m$ and stores $j = 1, \ldots, n$
- Each warehouse has a supply s_i and each store has a demand d_j
- Transporting inventory from warehouse i to store j costs c_{ij} per unit of the good
- What is the "flow" of inventory from the warehouses to the stores that fulfills the demands at minimum cost?

The optimal transport problem

- The flow from i to j is $x_{ij} \geq 0$
- The objective is to minimize the total cost: $\sum_{ij} c_{ij} x_{ij}$
- The supply and demand constraints are:

$$\sum_j x_{ij} \leq s_i \ \forall i = 1, \ldots, m$$

$$\sum_i x_{ij} \geq d_j \ \forall j = 1, \ldots, n$$

- NB the program is infeasible unless $\sum_i s_i \geq \sum_j d_j$

Dual programs

- For any LP (A, b, c) as in (P), we there is an associated **dual LP**

$$\min_{y \in \mathbb{F}^m} by \text{ s.t. } y \geq 0, yA \geq c \qquad \text{(D)}$$

- In the context of its dual, the program (P) is referred to as the **primal LP**
- On your pset, you'll verify that the dual of the dual is the primal, so that LPs do in fact come in pairs
- We refer to the primal-dual pair of LPs (P) and (D) as a **linear programming problem**
- Much of the theory of linear programming concerns the relationship between these two programs

Lagrangian duality

- The variables y are effectively Lagrange multipliers on the constraints $Ax \leq b$ in (P), and x are multipliers on the constraints $yA \geq c$ in (D)
- Indeed, consider the Lagrangian for (P):

$$\max_{x \geq 0} cx + y(b - Ax) = \max_{x \geq 0} by + (c - yA)x$$

- If $yA \ngeq c$, then the maximum can be made arbitrarily large by sending $x_i \to \infty$, where $yA^i < c_i$
- So, the value is finite only if $yA \geq c$

Lagrangian duality

- The variables y are effectively Lagrange multipliers on the constraints $Ax \leq b$ in (P), and x are multipliers on the constraints $yA \geq c$ in (D)
- Indeed, consider the Lagrangian for (P):

$$\max_{x \geq 0} cx + y(b - Ax) = \max_{x \geq 0} by + (c - yA)x$$

- If $yA \not\geq c$, then the maximum can be made arbitrarily large by sending $x_i \to \infty$, where $yA^i < c_i$
- So, the value is finite only if $yA \geq c$
- Similarly, consider

$$\min_{y \geq 0} by + (c - yA)x = \min_{y \geq 0} y(b - Ax) + cx$$

- The value is again finite iff $Ax \leq b$

Lagrangian duality

- The variables y are effectively Lagrange multipliers on the constraints $Ax \leq b$ in (P), and x are multipliers on the constraints $yA \geq c$ in (D)
- Indeed, consider the Lagrangian for (P):

$$\max_{x \geq 0} cx + y(b - Ax) = \max_{x \geq 0} by + (c - yA)x$$

- If $yA \ngeq c$, then the maximum can be made arbitrarily large by sending $x_i \to \infty$, where $yA^i < c_i$
- So, the value is finite only if $yA \geq c$
- Similarly, consider

$$\min_{y \geq 0} by + (c - yA)x = \min_{y \geq 0} y(b - Ax) + cx$$

- The value is again finite iff $Ax \leq b$
- NB The two programs have the same Lagrangian!
- In fact, we can interpret the LP problem as a zero-sum game, where the objective is $cx + by - yAx$, the maximizer chooses x, and the minimizer chooses y... more on this in a bit

Weak duality

Proposition (Weak duality)

If $x, y \geq 0$, $Ax \leq b$, $yA \geq c$, *then* $cx \leq by$.

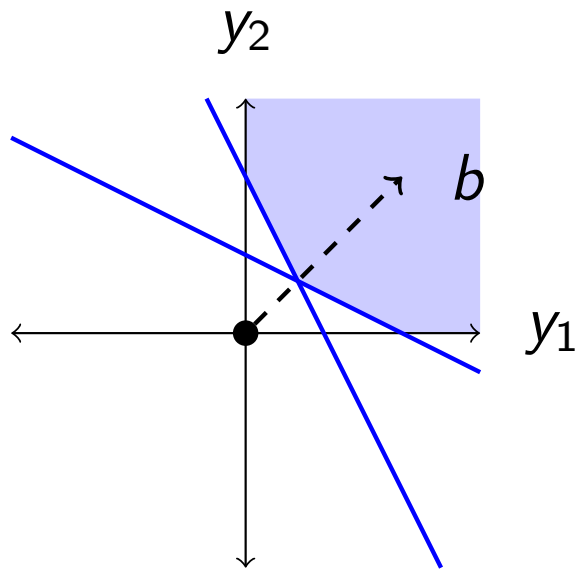
Proof.

$cx \leq (yA)x = y(Ax) \leq yb$ □

- So, any **feasible** solution of the dual gives us an upper bound on the value of the primal, and vice versa
- Relatedly, the optimal value of (P) must be weakly below that of (D) (that is, if optima exist!)
- Thus, if we solve both (P) and (D) simultaneously, then as we refine our solutions to each program, we also obtain refine bounds on how far we are from optimality

Toy example (2)

- What is the dual of our example LP?
- In this case, $c = (1, 1) = b$, $A = [2\ 1; 1\ 2]$
- $\min y_1 + y_2$ subject to $y_1 \geq 0$, $y_2 \geq 0$, $2y_1 + y_2 \geq 1$, $y_1 + 2y_2 \geq 1$



- Solution is again obvious; go to the SW corner, at $y_1 = y_2 = 1/3$!
- Optimal value is $2/3$... same as the primal!

Dual of the transportation problem

- We attach dual variables $p_j \geq 0$ to each of the demand constraints and $q_i \geq 0$ to each of the supply constraints
- The objective is to maximize

$$\sum_j p_j d_j - \sum_i q_i s_i$$

subject to

$$p_j - q_i \leq c_{ij}$$

- If we interpret the p_i as prices and q_j as costs, then this is saying that we are maximizing the net value of transportation, subject to the gain in value from moving good from i to j being at most c_{ij}, the cost of production!
- A meta theorem about linear programming is that given enough effort, dual variables can always be reinterpreted as prices; see DSS for further evidence

Saddle points

- A **saddle point** is a pair (x, y) that are feasible for (P) and (D), resp., and such hat $cx = by$
- An immediate implication of weak duality is that if we can find x and y that are feasible **and** $cx = by$, then x and y are both optimal for their respective problems
- So, saddle points "certify" the optimal value of the LP problem
- Much of our focus then is on understanding when saddle points exist and finding them

Complementary slackness

- A pair (x, y) satisfies **complementary slackness** if $(yA - c)x = 0$ and $y(b - Ax) = 0$
- In other words, for each column i, at least one of $x_i = 0$ and $yA^i = c_i$ holds, and for each row i, at least one of $y_i = 0$ and $A_i x = b_i$ holds

Theorem

A pair (x, y) is a saddle point if and only if it is feasible and satisfies complementary slackness.

- Proof:
  $by - cx = by - yAx + yAx - cx = y(b - Ax) + (yA - c)x$
- If (x, y) is feasible, then $y(b - Ax) \geq 0$ and $(yA - c)x \geq 0$, so that the duality gap is zero iff $y(b - Ax) = 0$ and $(yA - c)x = 0$
- So, (x, y) is a saddle point iff it is feasible and satisfies complementary slackness $\square$

The zero-sum game revisited

- Why are saddle points so named? Well, they correspond equilibria of the two-player zero-sum game with payoff

$$u(x, y) = cx + by - yAx,$$

where maximizer and minimizer choose $x \geq 0$ and $y \geq 0$, resp

The zero-sum game revisited

- Why are saddle points so named? Well, they correspond equilibria of the two-player zero-sum game with payoff

$$u(x, y) = cx + by - yAx,$$

where maximizer and minimizer choose $x \geq 0$ and $y \geq 0$, resp

- Indeed, if (x, y) is a saddle point, then by complementary slackness, the value of the game at these strategies is just the optimal value of the LP problem:

$$cx + \underbrace{y(b - Ax)}_{=0} = \underbrace{(c - yA)x}_{=0} + by$$

The zero-sum game revisited

- Why are saddle points so named? Well, they correspond equilibria of the two-player zero-sum game with payoff

$$u(x, y) = cx + by - yAx,$$

where maximizer and minimizer choose $x \geq 0$ and $y \geq 0$, resp

- Indeed, if (x, y) is a saddle point, then by complementary slackness, the value of the game at these strategies is just the optimal value of the LP problem:

$$cx + \underbrace{y(b - Ax)}_{=0} = \underbrace{(c - yA)x}_{=0} + by$$

- Moreover, since $b - Ax \geq 0$, for any $y' \geq 0$, we have that $y'(b - Ax) \geq 0$, so that $u(x, y') \geq cx = u(x, y)$
- Similarly, since $c - yA \leq 0$, if $x' \geq 0$, then $u(x', y) \leq by = u(x, y)$

Bounding the value of LPs

- Bottom line: The optimal multipliers minimize the value of the Lagrangian
- An immediate implication is that for **any** choice of $y \geq 0$, $\max_{x \geq 0} cx + yb - yAx$ is an upper bound on the optimal value of the primal
- Similarly, for any $x \geq 0$, $\min_{y \geq 0} cx + yb - yAx$ is a lower bound on the value of the dual
- As we will see, this fact can be quite useful:
  It makes it easy to get bounds on the optimal value of an LP, even if you don't know the exact optimal multipliers

Toy example (3)

- In light of this, a good way of finding a saddle point is to engineer (x, y) to satisfy complementary slackness
- In our example, let's conjecture that we have a saddle point with all variables strictly positive
- In this case, all of the constraints would **bind** (the dual variable is non-zero so the constraint has to hold as an equality), and we would need to have

$$
\begin{aligned}
2x_1 + x_2 &= 1 \\
x_1 + 2x_2 &= 1 \\
2y_1 + y_2 &= 1 \\
y_1 + 2y_2 &= 1
\end{aligned}
$$

- This system reduces to $x_1 = x_2 = y_1 = y_2 = 1/3$
- Clearly this solution is feasible, and we have $cx = 2/3 = yb$, so this is the solution!

Strong duality

- But do saddle points exist???
- Say that (P) is **feasible** if there exists $x \geq 0$ with $Ax \leq b$
- Moreover, (P) is **unbounded** if for any $M > 0$, there exists a feasible x with $cx > M$
- We extend these definitions in the obvious way to (D)

Theorem (Strong duality)

Given an LP problem (A, b, c), exactly one of the following is true:

(i) *Both* (P) *and* (D) *are feasible and a saddle point exists; a fortiori, the programs have the same value;*

(ii) *At least one of* (P) *and* (D) *is infeasible, and the other program is infeasible or unbounded*

- NB True in any ordered field!!! Don't need compactness/continuity for the existence of optima

Examples for case (ii)

- Consider the LP

$$\begin{aligned} \max_{x_1 \geq 0, x_2 \geq 0} \quad & c_1 x_1 + c_2 x_2 \\ \text{s.t.} \quad & x_1 - x_2 \leq -1 \\ & -x_1 + x_2 \leq -1 \end{aligned}$$

- Clearly infeasible, because

$$x_1 - x_2 \leq -1 \implies -x_1 + x_2 \geq 1$$

- The dual is

$$\begin{aligned} \min_{y_1 \geq 0, y_2 \geq 0} \quad & -y_1 - y_2 \\ \text{s.t.} \quad & y_1 - y_2 \geq c_1 \\ & -y_1 + y_2 \geq c_2 \end{aligned}$$

- With $c_1 = c_2 = 0$, the dual is unbounded (take $y_1 = y_2 \geq 0$ arbitrarily large)
- With $c_1 = c_2 = 1$, the dual is infeasible!

Why is strong duality important?

- In the words of Aeschylus, as Prometheus explains how he angered Zeus, so that he was punished by being chained to the rock in the Caucasus:

  PROMETHEUS: Yes, I caused mortals to cease foreseeing their doom.

  CHORUS: Of what sort was the cure that you found for this affliction?

  PROMETHEUS: I caused blind hopes to dwell within their breasts.

  CHORUS: A great benefit was this you gave to mortals.

- An approach to solving LP problems is to find a saddle point; this can be hard to do in general, and there is no guarantee that a search will be successful
- But strong duality gives us a blind hope that we can succeed, so that it is worth looking!

Farkas' Lemma

- We will prove the strong duality theorem via Farkas' lemma:

Lemma (Farkas)

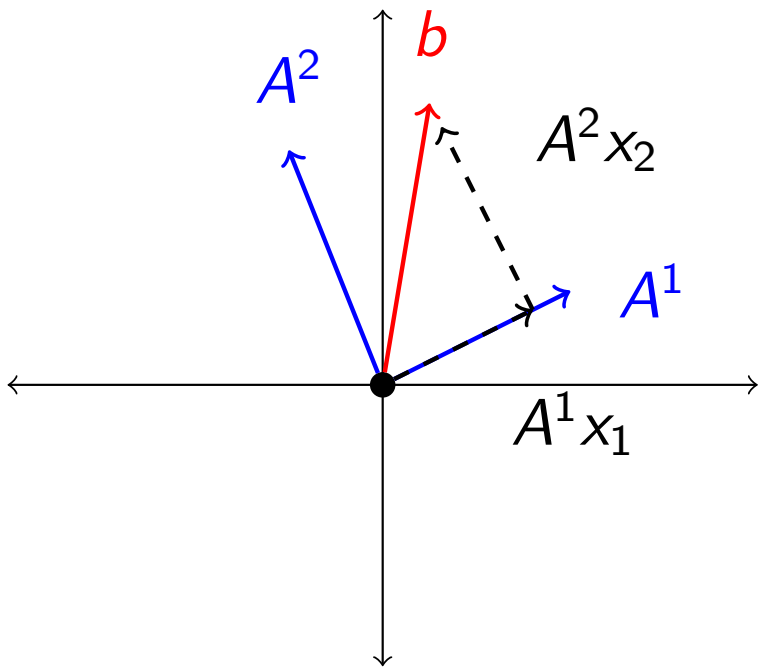
Given $A \in \mathbb{F}^{m \times n}$ and $b \in \mathbb{F}^m$, exactly one of the following is true:

(i) *There exists $x \in \mathbb{F}^n$, $x \geq 0$, such that $Ax = b$;*

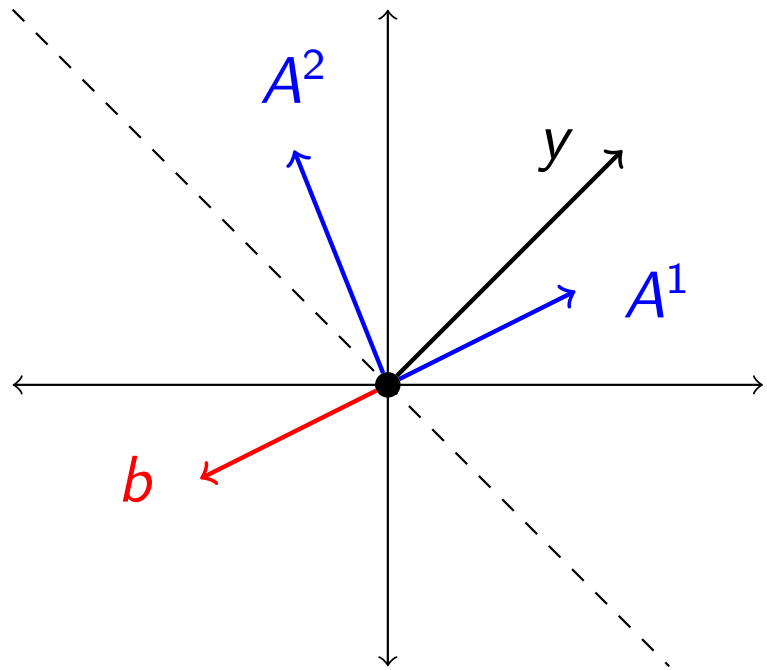
(ii) *There exists $y \in \mathbb{F}^m$ such that $yA \geq 0$ and $yb < 0$*

Interpretation and proof

- Either b is in the positive convex cone generated by the columns of A, or else there is a hyperplane that separates the columns of A from b



(a) $A^1x_1 + A^2x_2 = b$ has a solution.



(b) $([A^1\ A^2], b)$ is separated by y.

Proof of Farkas (1)

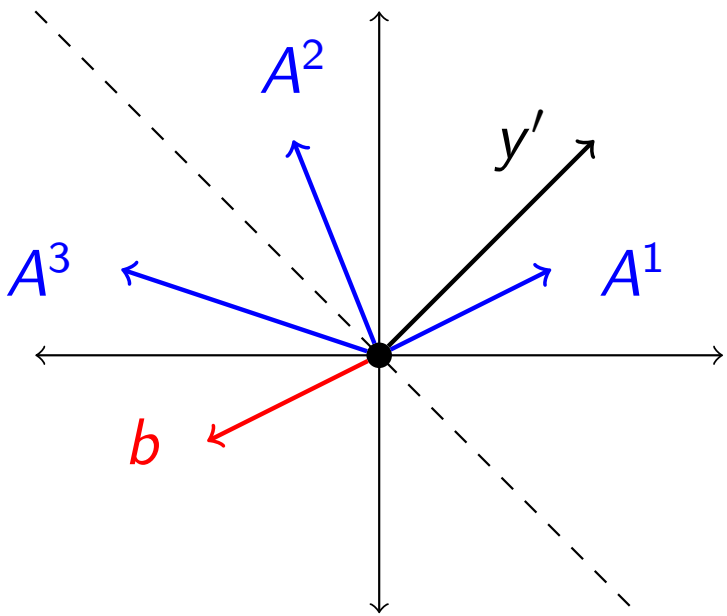
- First, at most one alternative can hold: If $Ax = b$, $x \geq 0$, and $yA \geq 0$, then $0 \leq yAx = yb$, so that $yb \not< 0$
- Now, we prove that $\neg(i)$ implies (ii), by induction on the number of columns (following Tucker (1956) or Gale (1960))
- If $n = 0$, then there are no columns, variables, and $Ax = b$ can hold iff $b = 0$; so if (i) does not hold, then $b \neq 0$, and we can take $y = -b$, and hence $yb = -bb < 0$
  (and trivially $yA = 0$ because there is nothing to sum)

Proof of Farkas (2)

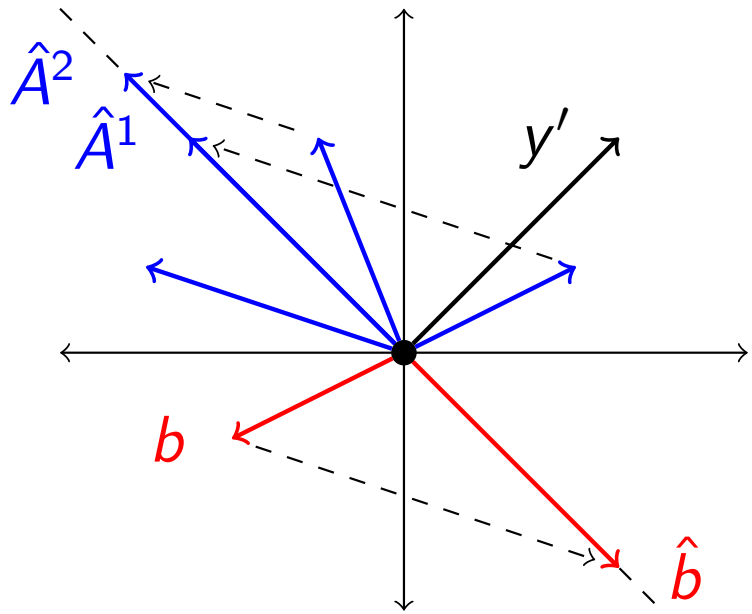
- Now the inductive step: Suppose Farkas holds for $n' < n$
- Let A' be the first $n-1$ columns of A; then exactly one of the following holds:
  - (i′) There is an $x' \geq 0$ such that $A'x' = b$;
  - (ii′) There is a y' such that $y'A \geq 0$ $y'b < 0$
- If (i′) holds, then so does (i), taking $x_i = x'_i$ for $i < n$ and $x_n = 0$, so we must be in case (ii′)
- If $y'A^n \geq 0$, then (ii) holds, and we are done
- So the remaining thorny case is if (ii′) holds but $y'A^n < 0$...

Proof of Farkas (3)

- High level idea: Project onto the subspace orthogonal to y, and then apply Farkas again
- We can always use A^n to "walk" to the level of b; the question is whether we can reach b using the other columns, in the subspace orthogonal to y'



(a) Before projecting.



(b) And after.

Proof of Farkas (4)

- Define the projected $\hat{A}$ and $\hat{b}$:

$$\hat{A}^i = A^i - \frac{y'A^j}{y'A^n}A^n, \;\; \hat{b} = b - \frac{y'b}{y'A^n}A^n$$

(net off from each vector how far we need to walk in the direction A^n to correct the level in the direction y')

- Now apply Farkas to $(\hat{A}, \hat{b})$, and we have two cases:

(i″) There exists $\hat{x}$ with $\hat{A}\hat{x} = \hat{b}$
Then take $x_i = \hat{x}_i$ for $i < n$ and
$x_n = \left(y'b - \sum_{i<n} y'A^i\hat{x} - i\right)/(y'A^n)$, so that

$$\begin{aligned} Ax &= \sum_{i<n} A^i\hat{x}_i + A^n(y'b - \sum_{i<n} y'A^i\hat{x}_i)/(y'A^n) \\ &= \hat{A}\hat{x} + A^n y'b/(y'A^n) \\ &= \hat{b} + A^n y'b/(y'A^n) = b \end{aligned}$$

and (i) holds, a contradiction

Proof of Farkas (5)

(ii″) There exists $\hat{y}$ with $\hat{y}\hat{A} \geq 0$, $\hat{y}\hat{b} < 0$

- In this case, let

$$y = \hat{y} - y' \frac{\hat{y}A^n}{y'A^n}$$

- This new direction strictly separates the component orthogonal to A^n:

$$yA^n = \hat{y}A^n - y'A^n \frac{\hat{y}A^n}{y'A^n} = 0$$

- Moreover,

$$yA^i = \hat{y}A^i - y'A^i \frac{\hat{y}A^n}{y'A^n} = \hat{y}\hat{A}^i \geq 0 \ \forall i < n$$

$$yb = \hat{y}b - y'b \frac{\hat{y}A^n}{y'A^n} = \hat{y}\hat{b} < 0$$

so that (ii) holds, as desired $\square$

Alternate form of Farkas

- I like the version of Farkas just introduced because it has a natural geometric interpretation
- There are many ways to rearrange the Farkas alternative, and here is one that will be immediately useful:

Lemma

Given (A, b), exactly one of the following holds:

(i) *There exists $x \geq 0$ such that $Ax \leq b$;*
(ii) *There is a $y \geq 0$ such that $yA \geq 0$ and $yb < 0$*

- Proof: Let $z \in \mathbb{R}^m$, I is the $n \times n$ identity matrix, then Farkas applied to $([A\ I], b)$ implies exactly one of the following is true
  - (i') There exist $x, z \geq 0$ and $[A\ I](x, z) = b$; or
  - (ii') There exist y s.t. $yb < 0$ and $y[A\ I] \geq 0$
- In case (i′), we have $x \geq 0$ and $Ax \leq b$, and in case (ii′), we have $yA \geq 0$, $y \geq 0$, $yb < 0$ $\square$
- z are sometimes called **slack variables**

Another interpretation of Farkas

- The feasibility of $x \geq 0$ and $Ax \leq b$ is **decidable** via a finite procedure, as shown by Fourier-Motzkin elimination:
  - Pick a variable, say x_1; every equation involving x_1 can be rearranged to either $x_1 \leq a(x_2, \ldots, x_n) + b$ or $x_1 \geq a(x_2, \ldots, x_n) + b$; call these class L and class G
  - Now replace x_1 with more inequalities, of the form

    $$a(x_2, \ldots, x_n) + b \leq a'(x_2, \ldots, x_n) + b'$$

    for every (a, b) in class G and (a', b') in class L (where $x_1 \geq 0$ is in class G); clearly, this new system is feasible iff the original system is feasible
  - Repeat to replace x_2, x_3, etc with more inequalities
- After n steps, every variable is eliminated, and we are left with a (very large but) finite collection of inequalities, to tell us if the original system is feasible
- If it is feasible, one can work backwards to find a feasible solution to the original system, and otherwise there is a procedure to find the unbounded direction

Proof of Strong Duality (1)

- There is a saddle point iff there exist $x, y \geq 0$ s.t. $Ax \leq b$, $-yA \leq -c$, and $by - cx \leq 0$, or equivalently:

$$\begin{bmatrix} A & 0 \\ 0 & -A^\top \\ -c^\top & b^\top \end{bmatrix} (x, y) \geq \begin{bmatrix} b \\ -c \\ 0 \end{bmatrix}$$

- If no saddle point exists, then (inequality) Farkas implies that there exist $\hat{x}, \hat{y}, \hat{z} \geq 0$ of lengths m, n, and 1, s.t.

$$(\hat{y}, \hat{x}, \hat{z}) \begin{bmatrix} A & 0 \\ 0 & -A^\top \\ -c & b \end{bmatrix} \geq 0, \quad (\hat{y}, \hat{x}, \hat{z}) \begin{bmatrix} b^\top \\ -c^\top \\ 0 \end{bmatrix} < 0$$

  i.e., $\hat{y}A - c\hat{z} \geq 0$, $\hat{z}b - A\hat{x} \geq 0$, and $\hat{y}b - \hat{x}c < 0$

- Observe that $0 = \hat{y}A\hat{x} - \hat{y}A\hat{x} \leq \hat{z}(\hat{y}b - \hat{x}c) \leq 0$, so $\hat{z} = 0$

Proof of Strong Duality (2)

- Thus, there exist $(\hat{x}, \hat{y})$ s.t. $\hat{y}A \geq 0$, $A\hat{x} \leq 0$, $\hat{y}b < \hat{x}c$
- Brooks and Reny (2021) refer to such a $(\hat{x}, \hat{y})$ as a **unbounded direction** (UD) for the LP problem
- We will prove that the existence of a UD implies that one program is infeasible and the other, if feasible, is unbounded
- Note that if $\hat{y}b \geq 0$, then $\hat{x}c > 0$, so that at least one of $\hat{y}b < 0$ and $\hat{x}c > 0$ is true

Proof of Strong Duality (3)

- The UD $(\hat{x}, \hat{y})$ satisfies $\hat{y}A \geq 0$, $A\hat{x} \leq 0$, and ($\hat{y}b < 0$ or $\hat{x}c > 0$ or both)
- Suppose $\hat{y}b < 0$:
  - If (P) is feasible, then there is an $x \geq 0$ such that $Ax \leq b$; but then $\hat{y}Ax \geq 0$ and so $\hat{y}b \geq 0$, a contradiction
  - Moreover, if (D) is feasible, then there is a $y \geq 0$ such that $yA \geq c$; but then $y + t\hat{y}$ is also feasible for all $t \geq 0$ since $(y + t\hat{y})A = yA + t\hat{y}A \geq c$, and the corresponding objective is $yb + t\hat{y}b$, which is unbounded below since $\hat{y}b < 0$
- The case where $c\hat{x} > 0$ is analogous and is on your problem set! $\square$

Aside on theorems of the alternative

- There are all sorts of "theorems of the alternative," e.g., Farkas, Tucker, strong duality, minimax
- In my view, LP duality is the most general and easiest to use
- If we take the primal to be maximization, and dual to be minimization, then there are easy rules to remember for going back and forth that follow the logic of the envelope theorem:

Primal	Dual
max	min
$x_i \geq 0$	$yA^i \geq c_i$
x_i free	$yA^i = c_i$
$x_i \leq 0$	$yA^i \leq c_i$
$A_i x \leq b_i$	$y_i \geq 0$
$A_i x = b_i$	y_i free
$A_i x \geq b_i$	$y_i \leq 0$

- Many theorems of the alternative can be easily derived as special cases of strong duality

Lagrangian relaxations

- Suppose that (x, y) is a saddle point
- Consider the alternative LP, for any $\lambda > 0$:

$$\max cx \text{ s.t. } x \geq 0,\ \lambda yAx \leq \lambda yb \qquad \text{(R)}$$

- Any x that was feasible for the original LP will also be feasible for this "relaxation", so it must have a weakly higher value
- But in fact, the two LPs must have the same value!
- Why? Well, (x, λ^{-1}) is a saddle point: the dual of (R) is

$$\min z\lambda yb \text{ s.t. } z \geq 0, z\lambda yA \geq c$$

and if we take $z = \lambda^{-1}$, then feasibility follows from that of y, and the value is yb

- What is the bottom line? If we aggregate constraints with weights that are proportional to the optimal Lagrange multipliers, then the value of the program remains the same; this elementary observation will be useful to us in the sequel

Solving LPs numerically

- An important feature of LPs (and a recurring theme of this lecture) is that it is relatively easy to verify a solution by exhibiting a saddle point
- An important strategy for finding a saddle point is to simply compute it numerically, and then analytically describe the solution
- We will discuss how to do this repeatedly in this course!
- Fortunately, there are now numerous high quality methods and implementations for solving LPs, which you are putting to good use on your problem set
- It is useful to talk in a bit more detail about how these algorithms work, so you can interpret the solution

Basic solutions

- The oldest and one of the best methods for solving LPs is the **simplex algorithm**, due to Dantzig, which finds a **basic solution**
- For this discussion, it is useful to formulate the LP as

$$\max cx \text{ s.t. } x \geq 0, Ax = b$$

- A **basic solution** of an LP is one for which we select a subset of the variables to be non-zero, and there is a unique solution to the corresponding subsystem
- In particular, fix a **basis** $I \subseteq \{1, \ldots, n\}$, and let A^I be the matrix consisting of columns in I
- If A^I is full rank, then the system $A^I x_I = b$ has a unique solution where only variables x_i for $i \in I$ are non-zero
- We refer to this as the basic solution corresponding to I

Sufficiency of basic solutions

- If an optimum exists, then there's a basic optimal solution (Follows from a similar logic as Caratheodory's theorem)
- Why? Let x be optimal and has a minimal number of positive entries (among optimal solutions)
- Suppose the coordinates x_i for $i \in I$ are strictly positive
- If x is not basic, then there is another $x' \geq 0$ such that $Ax' = b$, $x' \neq x$, and $x'_i > 0$ only if $x_i > 0$
- Notice that $\hat{x} = x + t(x' - x)$ also satisfies $A\hat{x} = b$, and for t small, $\hat{x} \geq 0$
  - $cx' \neq cx$, then $\hat{x}$ will be a strict improvement for $t \neq 0$ small, contradicting optimality of x
  - Thus, $cx' = cx$; but then $\hat{x}$ is also optimal for t small; we can move in some direction until some coefficient hits zero (because $x \neq x'$), which contradicts x having the fewest number of positive coordinates

The combinatorial natural of solutions & simplex algorithm

- Equivalently: Basic solutions are extreme points of the feasible set; since the objective is linear, there is an extreme point which is optimal
- The problem of finding an optimal basic solution is **combinatorial**, since we have to search through the (finitely) many bases
- The simplex algorithm operates by iterative "pivoting" between bases until an optimum is found:
  - Start from some incumbent basic feasible solution
  - Adding a new variable to the basis that increases the objective, drive out some existing variable
  - Repeat until an optimum is reached

Pivoting

- Take some x_i not currently in the basis, so $x_i = 0$
- We "introduce" x_i by making it positive, and adjusting all other variables to maintain feasibility:

$$A^I x_I + A^i x_i = b \iff x_I = (A^I)^{-1}(b - A^i x_i)$$

- As x_i increases, either some variable $j \in I$ eventually hits zero (and we've reached a new basic solution with basis $I \cup \{i\} \setminus \{j\}$), or the solution can increase without bound (in case we are moving along an extreme ray of the feasible set)

Reduced costs

- But what is the effect on the objective?
- The objective as a function of the entering variable x_i is

$$c_I x_I + c_i x_i = c_I (A^I)^{-1} b + (\underbrace{c_i - c_I (A^I)^{-1} A^i}_{\equiv r_i}) x_i$$

- The quantity r_i is the **reduced cost** for variable i
- The simplex algorithm operates by iteratively introducing a new variable into the basis for which $r_i > 0$
- Provided the LP is feasible, this procedure will either converge in finitely many steps to an optimum, or it will identify an unbounded ray
- The reason is that there are only finitely many basic solutions, and at every iteration, the value must strictly increase

Cold starting the simplex algorithm

- But where does the pivoting procedure start? How do you find an initial basic feasible solution?
- The most straightforward approach is to just add variables to the LP, so that it becomes $Ax + y - z = b$, where y and z are non-negative but otherwise unconstrained
- This LP is always feasible with $x = 0$ and $y - z = b$
- Then solve a first stage LP where the objective is to minimize $\Sigma y + \Sigma z$
- This "drives out" the artificial variables y and z, and the original LP is feasible iff the optimal value of the first stage is zero, and the optimum corresponds to a feasible solution to the original LP

Degeneracy

- This discussion presumes that the solution to $A^I x_I = b$ has all variables strictly positive, so that there is a one-to-one mapping between basic solutions and bases
- If this is the case, the LP is **non-degenerate**
- If the LP is degenerate, then there can be more than one basis corresponding to a given basic solution, and the pivoting procedure can lead to cycling
- Fortunately, various methods have been developed to address degeneracy, including introducing and removing perturbations to the coefficients, so this is not an issue we really need to worry about

Drawbacks of the simplex method

- Simplex is incredibly powerful and often quite fast
- Also, the combinatorial structure of basic solutions is convenient if, for example, you want to resolve the problem with a different objective
- But it has two irritating attributes:
  - (i) It is very slow on certain kinds of optimization problems; in particular, the worst-case runtime is exponential in the size of the problem
  - (ii) More importantly for our purposes, simplex finds an optimal basic solution, meaning that if there are multiple optimal solutions, it will force variables to be zero (in a somewhat arbitrary way) to pin down an optimal basis; this can obscure the essential structure of the solution, meaning, which variables **have** to be zero in an optimal solution

Interior point methods

- As an alternative, there are various "interior point" or "barrier" methods for solving LPs
- Basic idea: transform the optimization problem with constraints into an unconstrained optimization problem, by replacing the constraints with a penalty:

$$\max_{x \geq 0} cx + \mu \sum_i \log \left(b_i - A^i x \right)$$

- Starting with a large μ, we iteratively take a step of an unconstrained optimization algorithm (e.g., some form of gradient descent or Newton's method), and then decrease the penalty μ
- As long as $\mu \to 0$, this method will converge to an optimal solution of the original problem

Advantages of the interior point

- It was famously shown by Karmarkar that the interior-point algorithm runs in polynomial time (as a function of the size of the problem in bits and the error in the approximation) and is often efficient in practice
  (Ellipsoid algorithm was previously known to be polynomial but is inefficient in practice)
- (Aside: It is unknown whether there are **strongly** polynomial time algorithms for LPs, meaning that runtime is polynomial in the dimension (m, n), although the answer is yes for certain LPs, e.g., the transportation problem)
- Also, because the sequence approaches the optimum from inside the feasible set, it does not **necessarily** result in as many binding constraints as does the simplex algorithm (and in my experience it will result in many fewer binding constraints)
- This is important, because what we really want to know is: Which variables **have** to be zero and which constraints **have** to hold as equalities

Implementation and practical considerations

- On your problem set, I have asked you to use Python and Gurobi to solve LPs numerically, and plot the solutions
- Gurobi by default will solve an LP using a variety of algorithms
- It also solves both primal and dual simultaneously, so that we can verify how far we are from a solution
- Gurobi (and other commercial LP solvers like CPLEX) will also "cross over" from barrier to simplex
- I generally force Gurobi to use barrier and turn off the crossover
- Of course, solving the LP is only one half of the task; the other is visualizing the solution and extracting useful information about the optimal basis and the functional form of the saddle point

Simulation and approximation

- For small LPs, the software will often do a very convincing job of finding an exact optimum (as in our toy example)
- For larger LPs, it gets a bit murkier; computers have limited numerical precision (on the order of 2^{-52} 10^{-15} for many modern 64 bit computers)
- This is not as hard to approach as you might think, especially when using discrete models as an approximation of continuous ones
- Programs like Gurobi judge whether complementary slackness is satisfied up to some numerical tolerance
- So, when interpreting the solution, you may need to remember that the algorithm cannot solve the program exactly, and especially when extrapolating to continuous models, you will have to make some conceptual leaps about the form of the solution

Rest of the course

- We will now go on a journey through several topics in information economics, game theory, and mechanism design, largely relying on tools from linear programming
- For the models we consider, there are accompanying problem sets written in Python, in which you will "simulate" the model, and connect the numerical results in the simulation to the theoretical results and arguments in lecture
- For your final project, you will propose a **new model**, most likely in mechanism design or robust predictions
- You will have to write code to simulate the model and guess the analytical structure of the solution
- The last couple of weeks of the course will be in-class presentations of your preliminary findings
- My goal is to help you get started on a research project!