15-780 – Graduate Artificial Intelligence: Linear programming

J. Zico Kolter (this lecture) and Nihar Shah
Carnegie Mellon University
Spring 2020

Outline

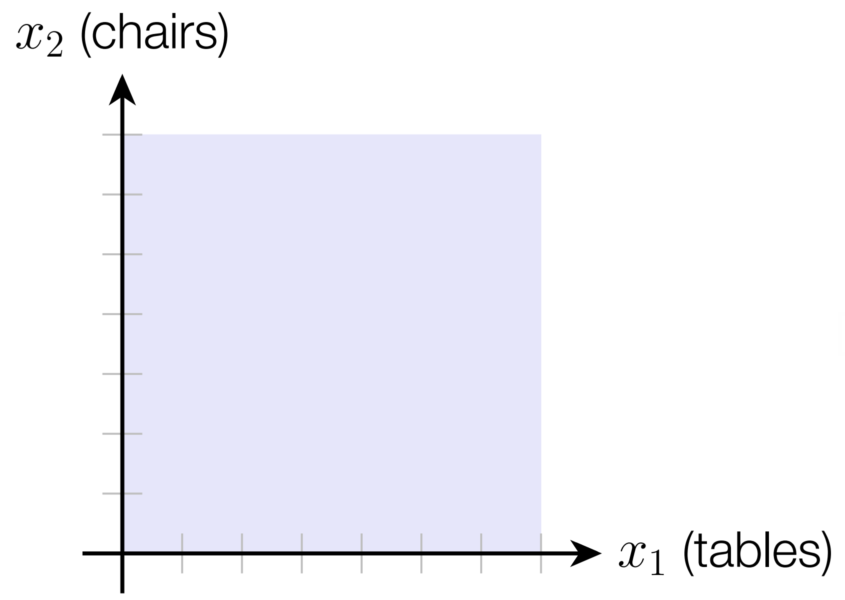
Outline

Example: manufacturing

A large factory makes tables and chairs. Each table returns a profit of $200 and each chair a profit of $100. Each table takes 1 unit of metal and 3 units of wood and each chair takes 2 units of metal and 1 unit of wood. The factory has 6K units of metal and 9K units of wood. How many tables and chairs should the factory make to maximize profit?

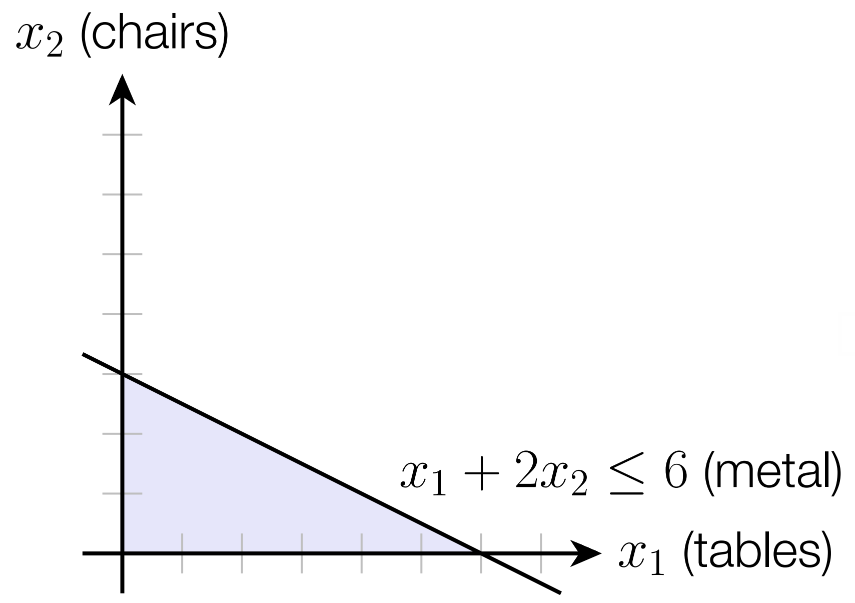
Example: manufacturing

A large factory makes tables and chairs. Each table returns a profit of \$200 and each chair a profit of \$100. Each table takes 1 unit of metal and 3 units of wood and each chair takes 2 units of metal and 1 unit of wood. The factory has 6K units of metal and 9K units of wood. How many tables and chairs should the factory make to maximize profit?



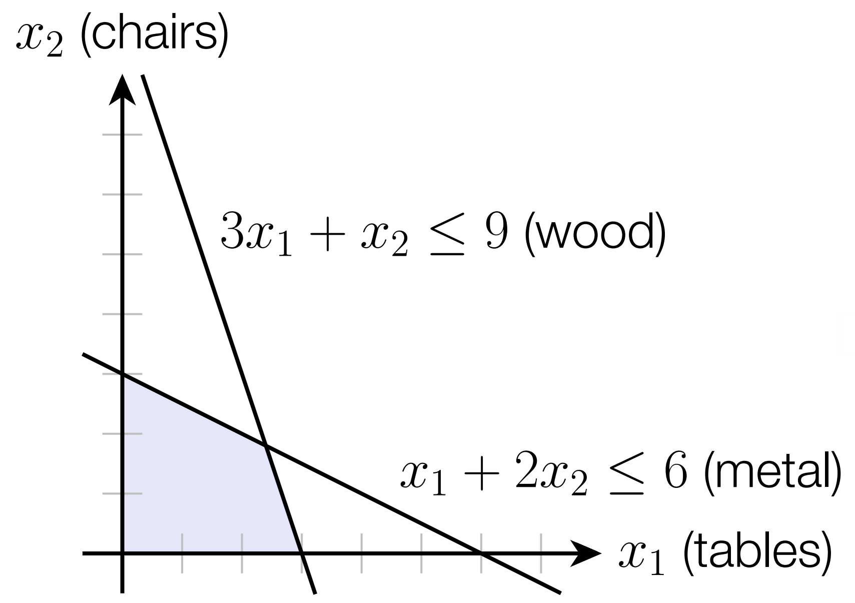
Example: manufacturing

A large factory makes tables and chairs. Each table returns a profit of $200 and each chair a profit of $100. Each table takes 1 unit of metal and 3 units of wood and each chair takes 2 units of metal and 1 unit of wood. The factory has 6K units of metal and 9K units of wood. How many tables and chairs should the factory make to maximize profit?



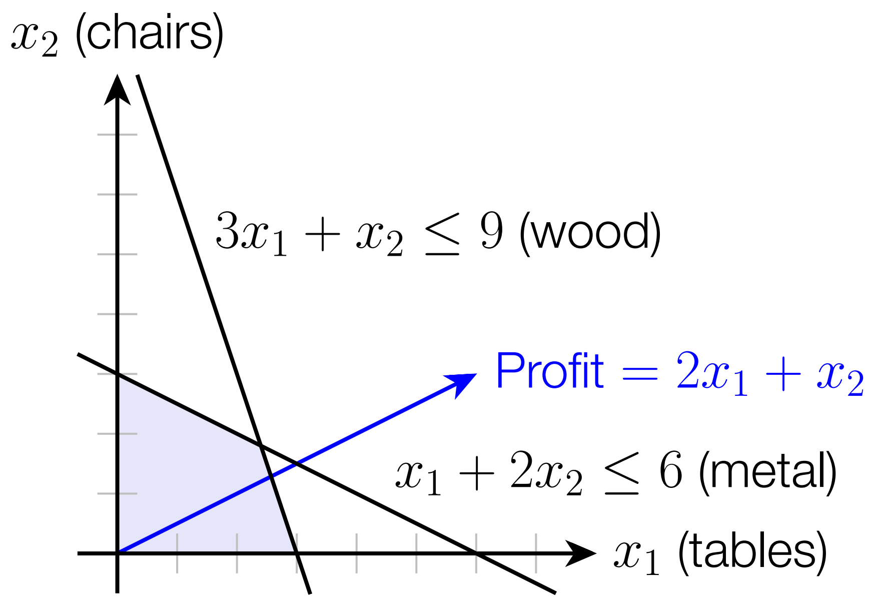
Example: manufacturing

A large factory makes tables and chairs. Each table returns a profit of $200 and each chair a profit of $100. Each table takes 1 unit of metal and 3 units of wood and each chair takes 2 units of metal and 1 unit of wood. The factory has 6K units of metal and 9K units of wood. How many tables and chairs should the factory make to maximize profit?



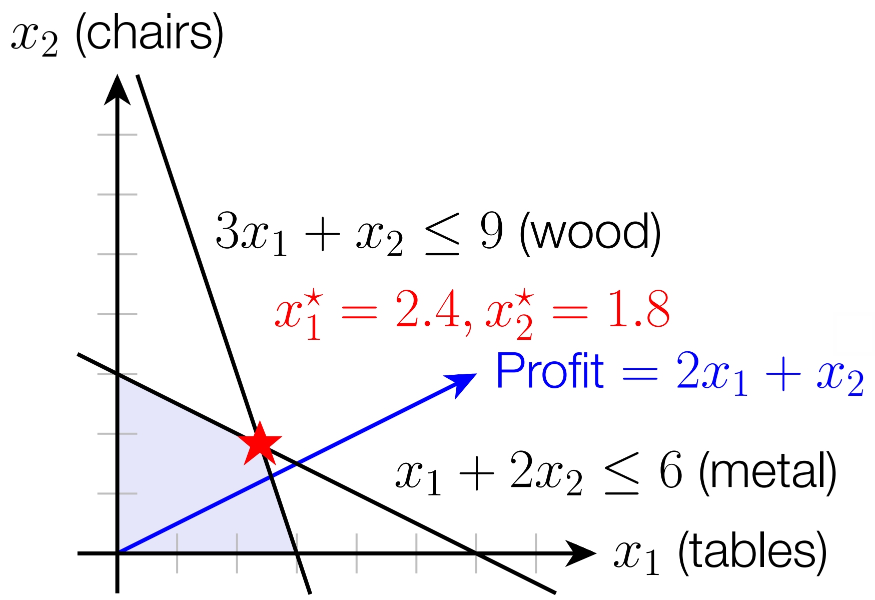
Example: manufacturing

A large factory makes tables and chairs. Each table returns a profit of \$200 and each chair a profit of \$100. Each table takes 1 unit of metal and 3 units of wood and each chair takes 2 units of metal and 1 unit of wood. The factory has 6K units of metal and 9K units of wood. How many tables and chairs should the factory make to maximize profit?



Example: manufacturing

A large factory makes tables and chairs. Each table returns a profit of \$200 and each chair a profit of \$100. Each table takes 1 unit of metal and 3 units of wood and each chair takes 2 units of metal and 1 unit of wood. The factory has 6K units of metal and 9K units of wood. How many tables and chairs should the factory make to maximize profit?



x_2 (chairs)
$3x_1 + x_2 \leq 9$ (wood)
$x_1^\star = 2.4, x_2^\star = 1.8$
Profit $= 2x_1 + x_2$
$x_1 + 2x_2 \leq 6$ (metal)
x_1 (tables)

Many applications

Finding optimal strategy in zero-sum two player games

Finding most probable assignment in probabilistic models

Finding solution in Markov decision processes

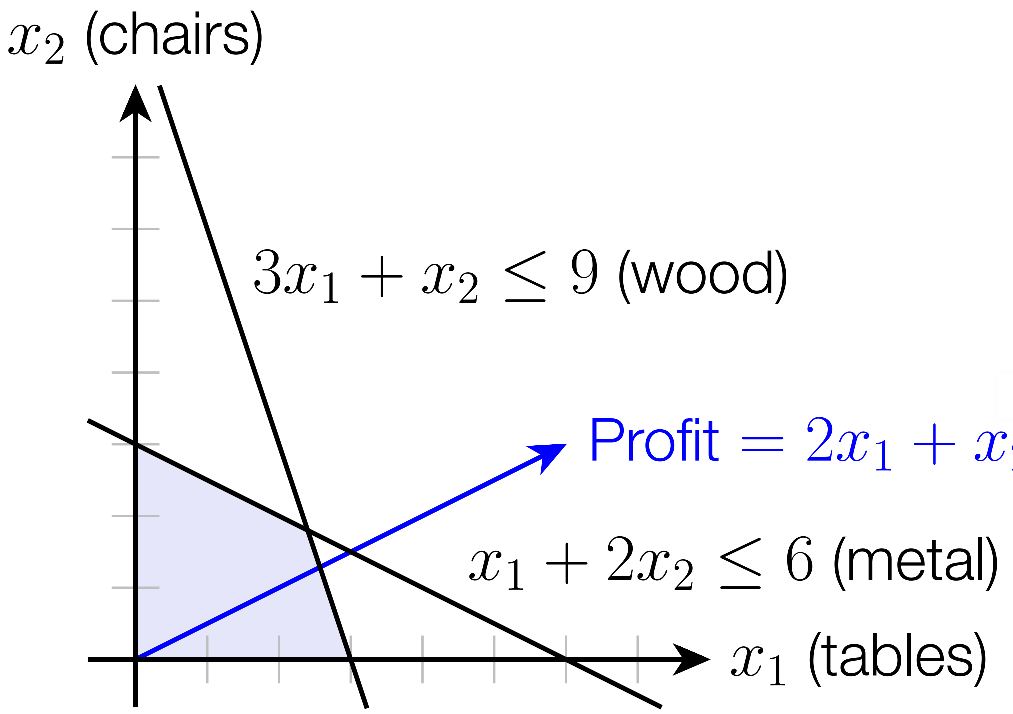
Min-cut / max-flow network problems

Applications: economic portfolio optimization, robotic control, scheduling generation in smart grids, many many others

Outline

Example manufacturing

We can write our manufacturing problem formally as:



$$
\begin{array}{ll}
\underset{x_1, x_2}{\text{maximize}} & 2x_1 + x_2 \\
\text{subject to} & 3x_1 + x_2 \leq 9 \\
& x_1 + 2x_2 \leq 6 \\
& x_1, x_2 \geq 0
\end{array}
$$

Inequality form linear programs

Using linear algebra notation, we can write problems compactly as

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & c^T x \\ \text{subject to} \quad & Gx \leq h \end{aligned}$$

with optimization variable $x \in \mathbb{R}^n$, problem data $c \in \mathbb{R}^n, G \in \mathbb{R}^{m\times n},\ h \in \mathbb{R}^m$ and where $\leq$ denotes elementwise inequality

A convex optimization problem (objective is affine, constraints are convex)

Our example:

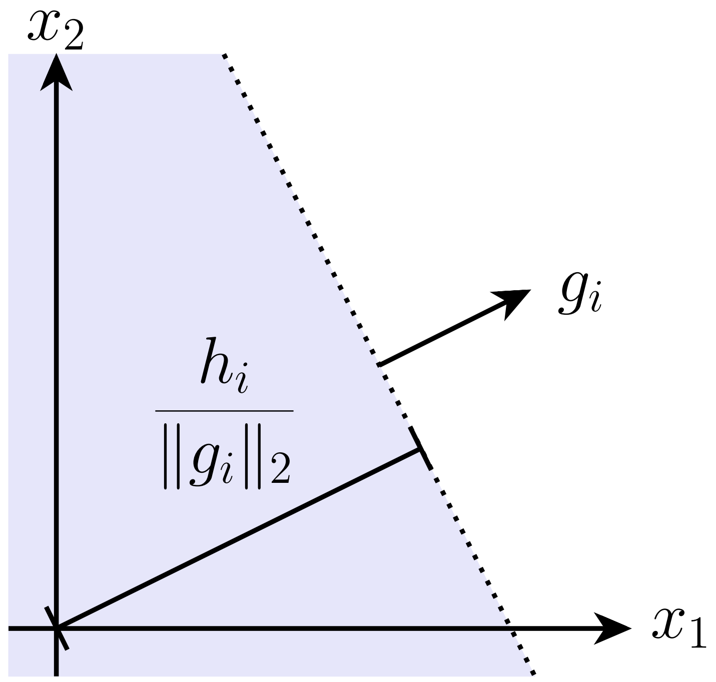
$$\begin{aligned} \underset{x_1,x_2}{\text{maximize}} \quad & 2x_1 + x_2 \\ \text{subject to} \quad & 3x_1 + x_2 \leq 9 \\ & x_1 + 2x_2 \leq 6 \\ & x_1, x_2 \geq 0 \end{aligned} \iff c = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, G = \begin{bmatrix} 3 & 1 \\ 1 & 2 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, h = \begin{bmatrix} 9 \\ 6 \\ 0 \\ 0 \end{bmatrix}$$

Geometry of linear programs

Consider a single row of the constraint matrix

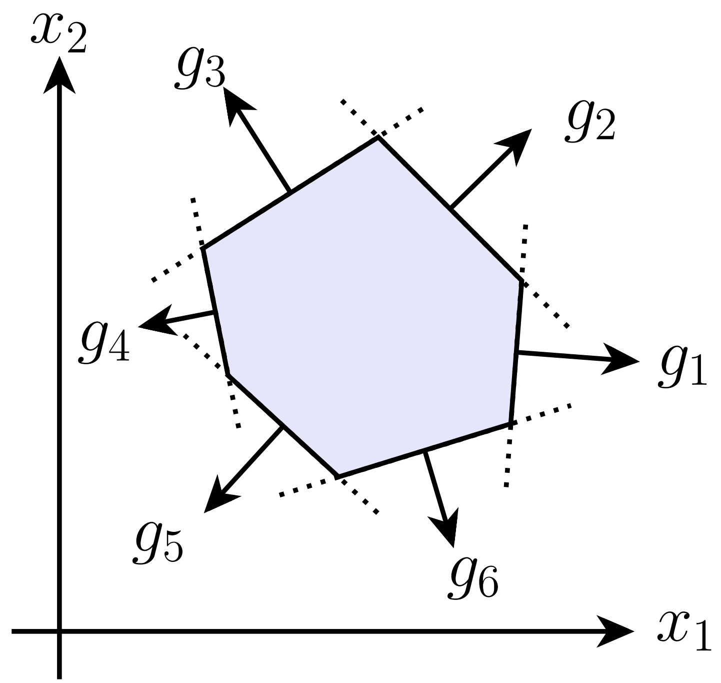
$$g_i^T x \le h_i$$

This represents a *halfspace* constraint



Linear polytope

Multiple halfspace constraints $Gx \leq h$ (i.e., $g_i^T x \leq h_i, i = 1, \ldots, m$) define what is called a *polytope*



So linear programming in equivalent to maximizing some direction over this polytope (note that optimum will always occur on a corner)

Poll: Number of corners in a polytope

Consider a polytope in n dimensional space, defined by $O(n)$ linear inequalities.
How many corners (vertices) of the polytope can there be?

1. $O(n)$
2. $O(n^2)$
3. $O(c^n)$
4. $O(n^n)$

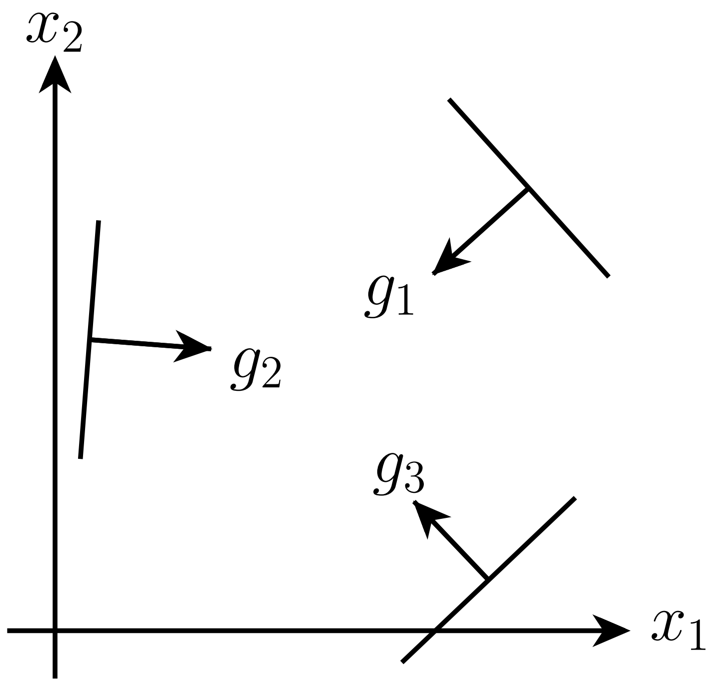
Poll: Number of inequalities in a polytope

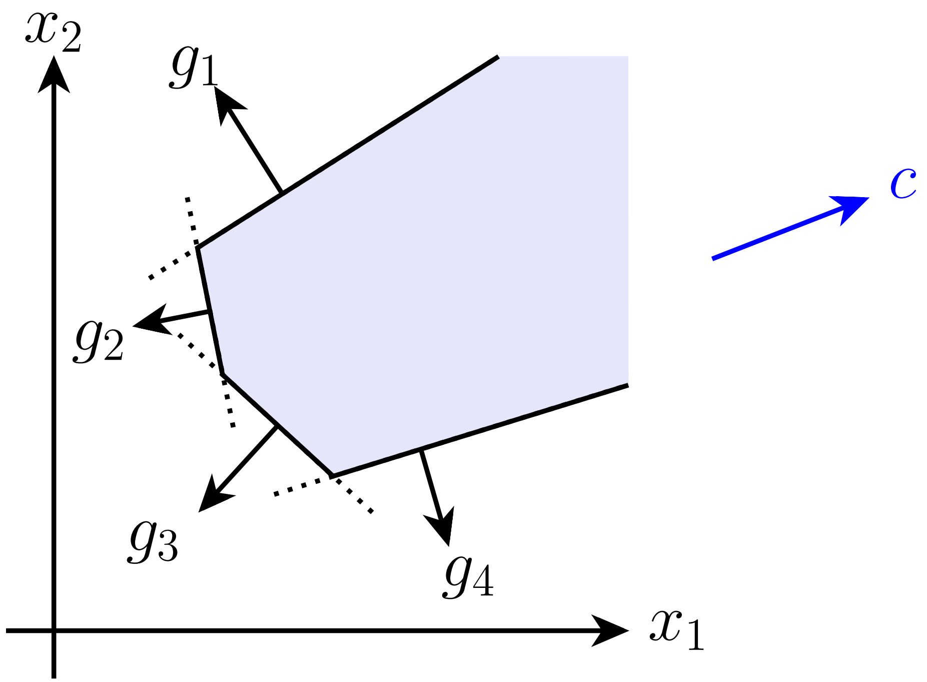
Consider a polytope in n dimensional space, with $O(n)$ corners (vertices). How many linear inequalities could we need to define the polytope?

1. $O(n)$
2. $O(n^2)$
3. $O(c^n)$
4. $O(n^n)$

Infeasible or unbounded polytopes

Polytopes may be infeasible or unbounded, correspond to have no solution or potentially an unbounded solution for linear program





Standard form linear programs

For the simplex algorithm, we will consider linear programs in an alternative form, known as *standard form*:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array}$$

with optimization variable $x \in \mathbb{R}^n$, and problem data $c \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n},\ b \in \mathbb{R}^m$ (note: m, n are not related to previous sizes)

Looks different, but it is straightforward to convert between inequality form and standard form by adding *slack variables*

$$g_i^T x \leq h_i \implies g_i^T x + s_i = h_i, s_i \geq 0$$

Can also separate non-negative variables in positive/negative part

Converting to standard form

Can convert our example problem to standard form:

$$\begin{aligned} \underset{x_1, x_2}{\text{maximize}} \quad & 2x_1 + x_2 \\ \text{subject to} \quad & 3x_1 + x_2 \leq 9 \\ & x_1 + 2x_2 \leq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \underset{x_1, x_2, x_3, x_4}{\text{minimize}} \quad & -2x_1 - x_2 \\ \text{subject to} \quad & 3x_1 + x_2 + x_3 = 9 \\ & x_1 + 2x_2 + x_4 = 6 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

$$c = \begin{bmatrix} -2 \\ -1 \end{bmatrix}, A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

Finding corners in polytope

In standard form we assume $n > m$ (plus some technical conditions like full row rank), so A is an underdetermined system of linear equations

To find solutions to subsets of these equations, we can select m columns from A, denoted $A_{\mathcal{J}}$ for some set $\mathcal{J} \subset \{1, \ldots, n\}$ with $|\mathcal{J}| = m$, and solve the resulting linear system

$$A_{\mathcal{J}} x_{\mathcal{J}} = b$$

(then set remaining entries of x to zero)

Solutions for which $x \geq 0$, correspond to corners on the polytope

Note: We'll use $A_{\mathcal{J}}$ to denote subselecting columns of A, A_j to denote the jth column of A, and $x_{\mathcal{J}}$ to denote subselecting element of x

Finding corners in polytope

Polytope from our example:

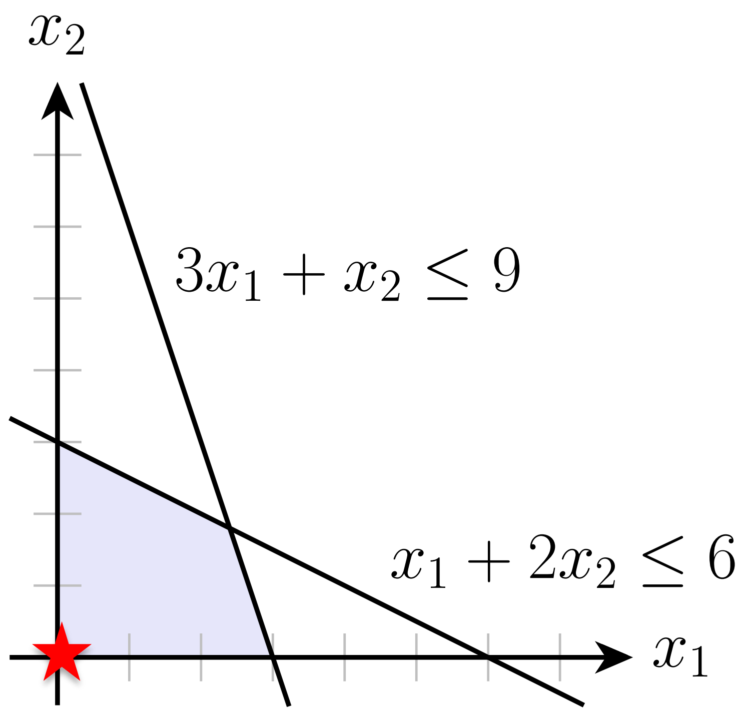
$$A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

$$\mathcal{J} = \{3,4\}$$

$$x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 9 \\ 6 \end{bmatrix} = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

$$x = \begin{bmatrix} 0 \\ 0 \\ 9 \\ 6 \end{bmatrix}$$



Finding corners in polytope

Polytope from our example:

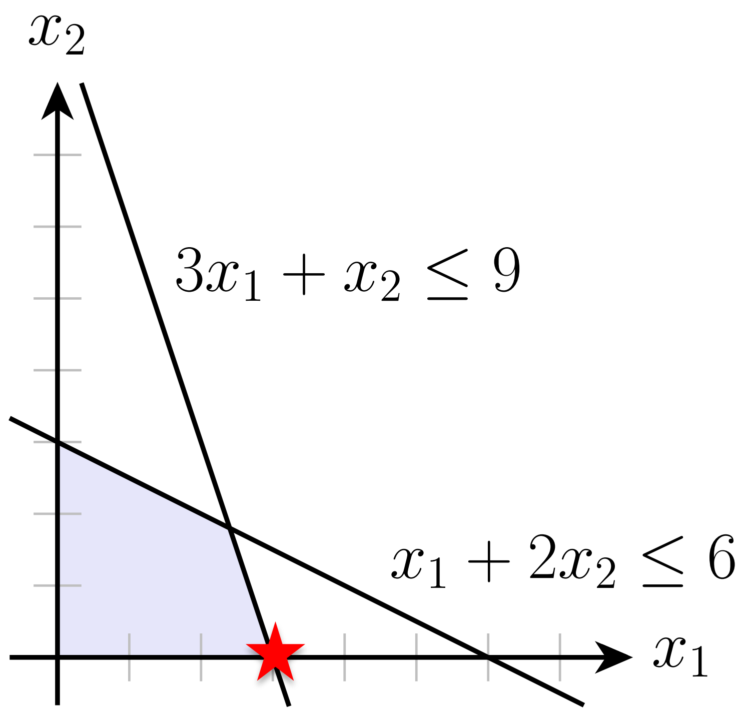
$$A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

$$\mathcal{J} = \{1{,}4\}$$

$$x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b$$

$$= \begin{bmatrix} 3 & 0 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 9 \\ 6 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$x = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 3 \end{bmatrix}$$



Finding corners in polytope

Polytope from our example:

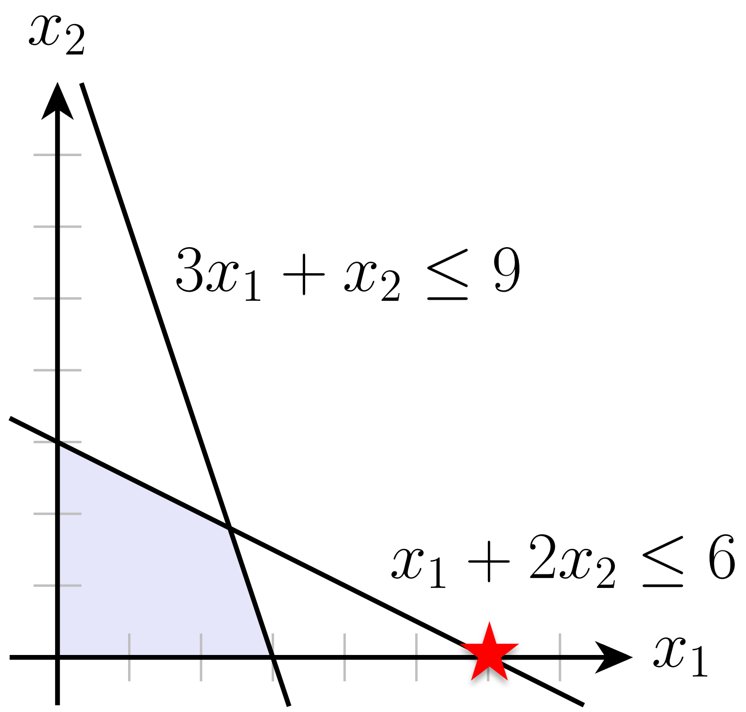
$$A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

$$\mathcal{J} = \{1,3\}$$

$$x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b$$

$$= \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 9 \\ 6 \end{bmatrix} = \begin{bmatrix} 6 \\ -9 \end{bmatrix}$$

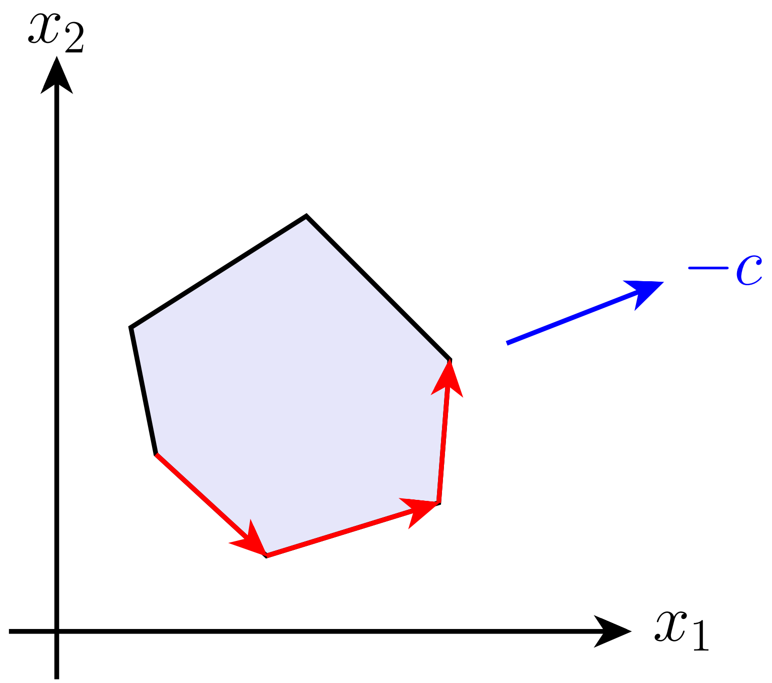
$$x = \begin{bmatrix} 6 \\ 0 \\ -9 \\ 0 \end{bmatrix}$$



Outline

Simplex algorithm

Basic idea of the simplex algorithm is to move along the edges of the polytope from corner to corner, in directions of decreasing cost



In worst case, move along an exponentially large number of corners, but typically much better in practice (the first "practical" algorithm for linear programming)

A single step of the simplex algorithm

Suppose we are optimizing the standard form linear program

$$\begin{aligned} \underset{x}{\text{minimize}} \;\; & c^T x \\ \text{subject to} \;\; & Ax = b \\ & x \geq 0 \end{aligned}$$

and suppose we have some initial feasible corner point x (i.e., we have a basis $\mathcal{J}$ such that $x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b \geq 0$)

(Seems like a big assumption, but we can relax this)

We would like to *adjust* this point so as to further decrease the cost, but how do we go about adjusting it?

A single step of the simplex algorithm (cont)

Suppose we want to adjust x by setting $x_j \leftarrow \alpha$ for some $j \notin \mathcal{J}$ (i.e., so $x_j = 0$ initially), in hopes of decreasing the objective

We cannot only change $x_j \leftarrow \alpha$, because this new point would not satisfy the linear equalities

$$A_{\mathcal{J}} x_{\mathcal{J}} = b \quad \Longrightarrow \quad A_{\mathcal{J}} x_{\mathcal{J}} + \alpha A_j \neq b$$

Instead, we need to adjust $x_{\mathcal{J}}$ by some $\alpha d_{\mathcal{J}}$ to ensure that the equality constraint still holds

$$\begin{aligned} A_{\mathcal{J}}(x_{\mathcal{J}} + \alpha d_{\mathcal{J}}) + \alpha A_j = b &\Longrightarrow \alpha A_{\mathcal{J}} d_{\mathcal{J}} = b \; - A_{\mathcal{J}} x_{\mathcal{J}} - \alpha A_j \\ &\Longrightarrow \alpha A_{\mathcal{J}} d_{\mathcal{J}} = -\alpha A_j \qquad (\text{because } A_{\mathcal{J}} x_{\mathcal{J}} = b) \\ &\Longrightarrow d_{\mathcal{J}} = -A_{\mathcal{J}}^{-1} A_j \end{aligned}$$

A single step of the simplex algorithm (cont)

Now suppose we adjust $x_{\mathcal{J}} \leftarrow x_{\mathcal{J}} + \alpha d_{\mathcal{J}}$ and $x_j \leftarrow \alpha$, how does this change the objective of our optimization problem?

$$c^T x \leftarrow c^T x + \alpha(c_j + c_{\mathcal{J}}^T d_{\mathcal{J}})$$

In other words, setting x_j to be α will increase objective by $\alpha \bar{c}_j$ where

$$\bar{c}_j = c_j - c_{\mathcal{J}}^T A_{\mathcal{J}}^{-1} A_j$$

Thus, as long as $\bar{c}_j$ is negative, it is a "good idea" to adjust x_j in this manner (if more than one $\bar{c}_j$ is negative, we could pick any)

If no $\bar{c}_j < 0$, we have found a solution!

A single step of the simplex algorithm (cont)

Final question: how big of a step $x_j \leftarrow \alpha$ should we take?

If all $d_{\mathcal{J}} \geq 0$, we are in “luck”, we can decrease the optimization objective arbitrarily without leaving the feasible region (i.e., an unbounded problem)

But if some element $d_i < 0$, for $i \in \mathcal{J}$, we can take at most a step of size:

$$x_i + \alpha d_i = 0 \implies \alpha = -x_i / d_i$$

So, take the biggest step we can while keeping x positive, i.e., find:

$$i^\star = \operatorname*{argmin}_{i \in \mathcal{J} : d_i < 0} -x_i / d_i$$

and take step such that $x_{i^\star} = 0$ (at this point, j enters $\mathcal{J}$ and $i^\star$ leaves)

Simplex algorithm

Given index set $\mathcal{J}$ such that $x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b \geq 0$

Repeat:

1. Find j for which $\bar{c}_j = c_j - c_{\mathcal{J}}^T A_{\mathcal{J}}^{-1} A_j < 0$ (if none exists, return x)
2. Compute step direction $d_{\mathcal{J}} = -A_{\mathcal{J}}^{-1} A_j$ and determine index to remove (or return unbounded if $d_{\mathcal{J}} \geq 0$)
$$i^\star = \operatorname*{argmin}_{i \in \mathcal{J} : d_i < 0} -x_i / d_i$$
3. Update index set: $\mathcal{J} \leftarrow \mathcal{J} - \{i^\star\} \cup \{j\}$

Illustration of simplex algorithm

Polytope from our example:

$$A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

$$\mathcal{J} = \{3,4\}$$

$$x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

$$\bar{c}_{1,2} = (-2, -1)$$

Choosing $j = 1$:

$$d_{\mathcal{J}} = -A_{\mathcal{J}}^{-1} A_1 = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$$

$$i^\star = \underset{i \in \{3,4\}}{\text{argmin}} \{3\colon 9/3\,, 4\colon 6/1\} = 3$$

$$\mathcal{J} = \mathcal{J} - \{3\} \cup \{1\}$$

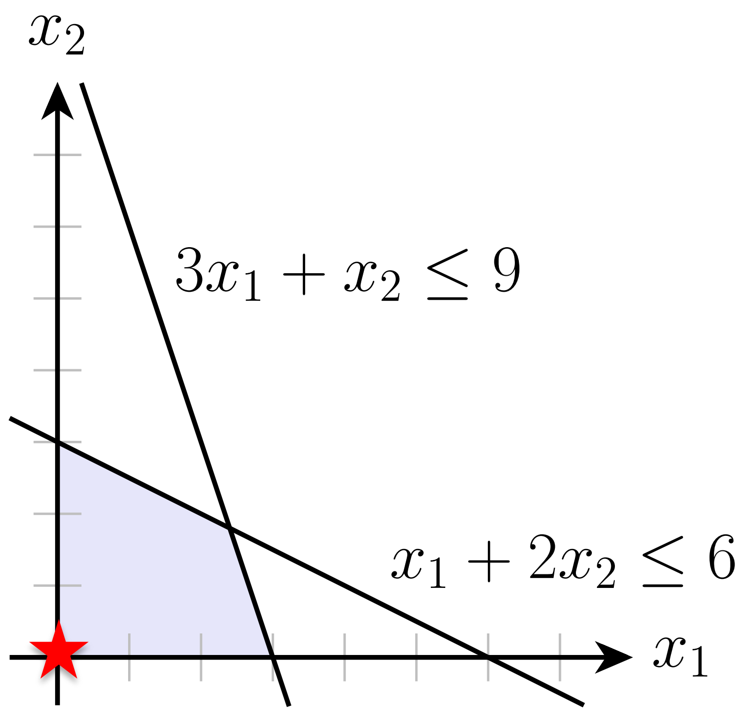


Illustration of simplex algorithm

Polytope from our example:

$$A = \begin{bmatrix} \boxed{3} & 1 & 1 & \boxed{0} \\ \boxed{1} & 2 & 0 & \boxed{1} \end{bmatrix}, b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

$$\mathcal{J} = \{1,4\}$$

$$x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\bar{c}_{2,4} = (-1/3, 2/3)$$

Choosing $j = 2$:

$$d_{\mathcal{J}} = -A_{\mathcal{J}}^{-1} A_1 = \begin{bmatrix} -1/3 \\ -5/3 \end{bmatrix}$$

$$i^\star = \underset{i \in \{1,4\}}{\operatorname{argmin}} \{1 : 9 \, , 4 : 9/5\} = 3$$

$$\mathcal{J} = \mathcal{J} - \{4\} \cup \{2\}$$

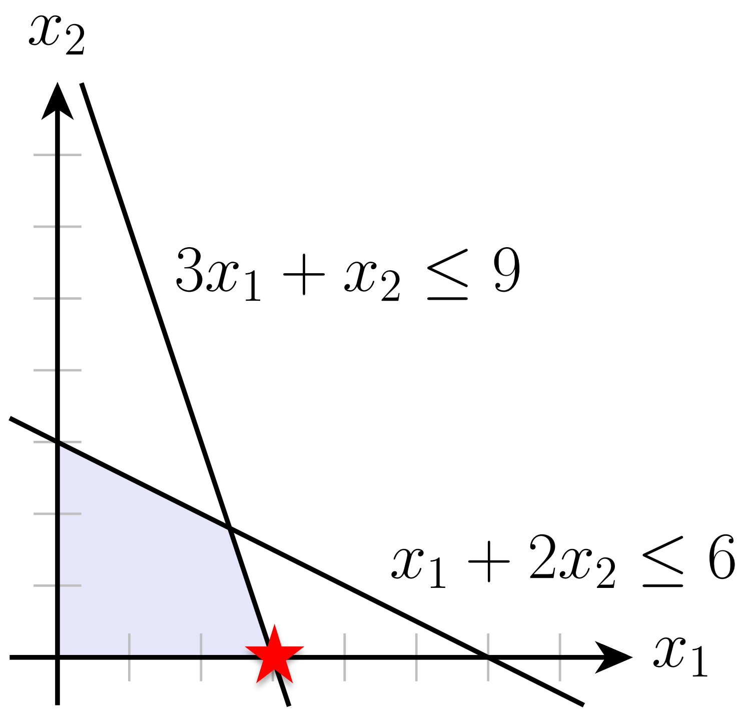


Illustration of simplex algorithm

Polytope from our example:

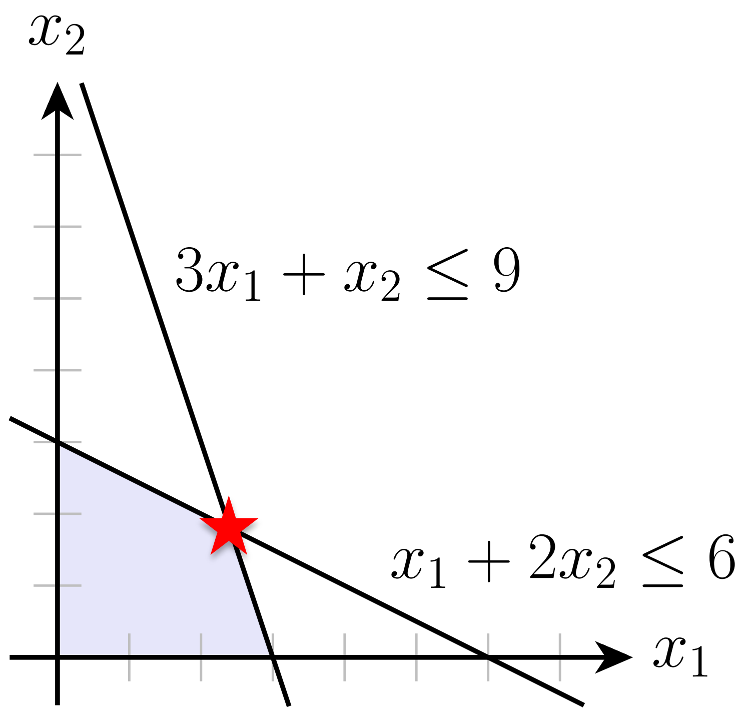
$$A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

$$\mathcal{J} = \{1,2\}$$

$$x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b = \begin{bmatrix} 2.4 \\ 1.8 \end{bmatrix}$$

$$\bar{c}_{3,4} = (0.6, 0.2)$$

Since all $\bar{c}_j$ are positive, we are done



Simplex solves linear programs

Theorem: the simplex algorithm is guaranteed to find a globally optimal solution to a linear program

Proof: (ignoring some possible degenerate cases)

If simplex returns, then it has found a point where we cannot improve the objective locally; since linear programs are convex, this is a global optimum

Because the objective of the simplex improves at each iteration, and since there are a finite (but exponential) number of vertices in the polytope, the algorithm must return after a finite number of steps

Poll: simplex complexity

What is the complexity of a *single iteration* of the simplex algorithm? (remember that matrix $A \in \mathbb{R}^{m \times n}$)

1. $O(mn)$
2. $O(m^2 n)$
3. $O(m^3 + mn)$
4. $O(m^3 + m^2 n)$
5. $O(m^2 + mn)$

Numerical considerations

Note that the above algorithm is described in terms of exact math

When implemented in floating point arithmetic, we'll often get entries like

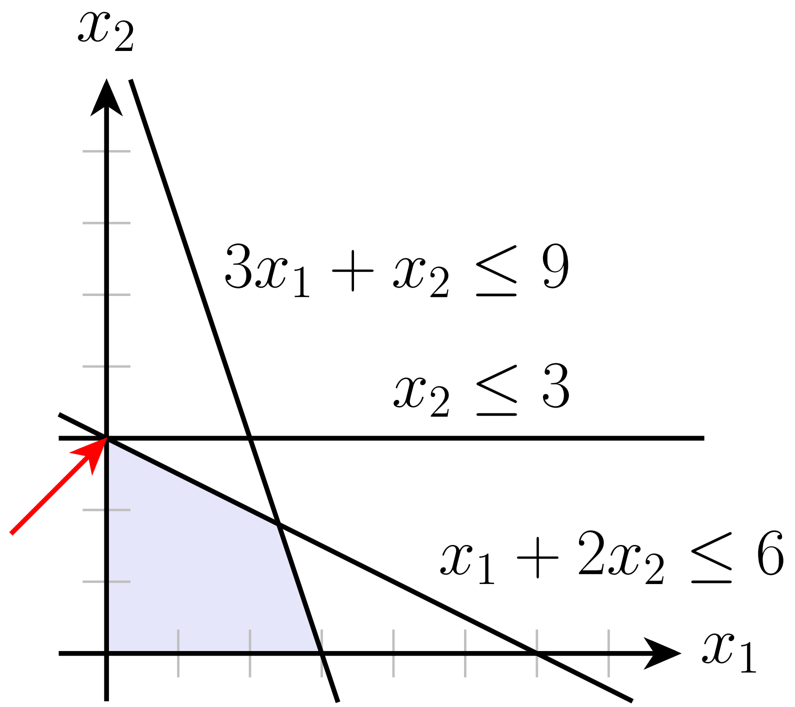
$$\bar{c}_j, d_i \in [-10^{-15}, 10^{-15}]$$

When comparing to zero, or comparing "ties" (see next slide), we need to account for this

In practice, set these near-zero elements to zero or compare to $\pm 10^{-12}$

Degeneracy

More than m constraints may intersect at a given point, at such a corner the simple solution will have $x_i = 0$ for some $i \in \mathcal{J}$



To make progress from such points, we may need to take a step size $\alpha = 0$ (i.e., remain at the same point, but switch which columns are in $\mathcal{J}$)

Handling degeneracy

Simplex will still work with zero step sizes, but we need to take care not to "cycle" repeatedly over the same indices

A similar issue can occur when determining which $i^\star$ exists the set $\mathcal{J}$ (i.e., if more than one x_i becomes zero at the same time)

A simple approach to fix these problems, *Bland's rule*

1. At each step, choose smallest j such that $\bar{c}_j < 0$
2. For variables x_i that could exit set $\mathcal{J}$ choose the smallest $i^\star$

Alternatively, perturb b by some small noise and then solve, then resolve with exact b and the optimal index set

Finding feasible solutions

We assumed an initial feasible point, i.e. $\mathcal{J}$ such that $A_{\mathcal{J}}^{-1} b \geq 0$

To find such a point (in cases where it is not easy to construct one, we introduce variables $z \in \mathbb{R}^m$ and solve auxiliary problem

$$\begin{aligned} \underset{x,z}{\text{minimize}} \;& 1^T z \\ \text{subject to } & Ax + z = b \\ & x, z \geq 0 \end{aligned}$$

where for all $b_i < 0$ we replace constraint $a_i^T x = b_i$ with $-a_i^T x = -b_i$

Initial feasible solution given by $x = 0, z = b$, and if final solution has $z = 0$, we have found a feasible solution to the original problem

Revised simplex

Simplex algorithm require inverting $A_{\mathcal{J}}$ at each iteration: naive implementation would re-invert matrix at every iteration

Revised simplex algorithm directly maintains/updates the inverse $A_{\mathcal{J}}^{-1}$

Key idea is the *Sherman-Morrison* inversion formula, for an invertible matrix $C \in \mathbb{R}^{n\times n}$ and vectors $u, v \in \mathbb{R}^n$

$$(C + uv^T)^{-1} = C^{-1} - \frac{C^{-1}uv^TC^{-1}}{1 + v^TC^{-1}u}$$

Each update *overwrites* some column $(A_{\mathcal{J}})_k$ with A_j, i.e.,

$$A_{\mathcal{J}} \leftarrow A_{\mathcal{J}} + (A_j - (A_{\mathcal{J}})_k)e_k^T$$

Simplex tableau

If you have been taught the simplex algorithm before (or if you read virtually any book on the subject), you will probably see tables that look like this:

	6.6	0	0	0.2	0.6
1	2.4	1	0	-0.2	0.4
2	1.8	0	1	0.6	-0.2

This is the *simplex tableau,* and it is just an organization of all the relevant quantities in the simplex algorithm:

	$-c^T x$	$\bar{c}^T = c^T - c_{\mathcal{J}}^T A_{\mathcal{J}}^{-1} A$
$\mathcal{J}$	$x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b$	$A_{\mathcal{J}}^{-1} A$

along with a set of operations for performing the updates (essentially doing the same this as the Sherman-Morrison formula)

Outline

Lagrangian duality

Duality is an extremely powerful concept in convex optimization in general (we consider it first for linear programs, but then highlight general case)

Given a linear program in standard form

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array}$$

we define a function called the *Lagrangian*, which has the form

$$\mathcal{L}(x, y, z) = c^T x + y^T (Ax - b) - z^T x$$

where $y \in \mathbb{R}^m$, $z \in \mathbb{R}^n$ are called *dual variables* for the constraints $Ax = b$ and $x \geq 0$ respectively

Min-max formulation

First note that

$$\max_{y,z\geq 0} \mathcal{L}(x,y,z) = \begin{cases} c^T x & \text{if } Ax = b, x \geq 0 \\ \infty & \text{otherwise} \end{cases}$$

Can write the original optimization problem (called the primal problem) as

$$\min_{x} \max_{y,z\geq 0} \mathcal{L}(x,y,z)$$

We are effectively using the min/max setup to express the same constraints as in the standard form problem

Alternatively, we could flip the order of the min/max to obtain what is called the dual problem

$$\max_{y,z\geq 0} \min_{x} \mathcal{L}(x,y,z)$$

Weak and strong duality

Denoting the optimal solutions to the primal and dual problems as $p^\star$ and $d^\star$ respectively, we immediately have the following (called *weak duality)*

$$d^\star = \max_{y,z\geq 0} \min_x \mathcal{L}(x,y,z) \leq \min_x \max_{y,z\geq 0} \mathcal{L}(x,y,z) = p^\star$$

(if we minimize over x first, and then maximize over y, z this is always larger than if we maximize over y, z first and then minimize over x)

A remarkable property: for linear programs, we actually have $p^\star = d^\star$ (called *strong duality*), and the simplex algorithm actually gives us solutions for both the primal *and* dual problems

Also crucial: *any feasible solution* to dual problem provides a *lower bound* on the optimal primal solution

Dual problem for standard form LP

For the standard form LP, note that the inner minimization is given by

$$\begin{aligned}\min_x \mathcal{L}(x,y,z) &= \min_x c^T x + y^T(Ax-b) - z^T x \\ &= \begin{cases} -b^T y & \text{if } A^T y + c - z = 0 \\ -\infty & \text{otherwise} \end{cases}\end{aligned}$$

Thus, we can write the dual problem as

$$\begin{array}{ll} \underset{y,z}{\text{maximize}} & -b^T y \\ \text{subject to} & A^T y + c = z \\ & z \geq 0 \end{array} \equiv \begin{array}{ll} \underset{y}{\text{maximize}} & -b^T y \\ \text{subject to} & -A^T y \leq c \end{array}$$

(a linear program in inequality form!)

Strong duality for LPs

Theorem: for linear programs, strong duality holds ($p^\star = d^\star$) and the simplex algorithm gives solutions $x^\star$ and $y^\star$

Proof: When simplex algorithm terminates, we have $x^\star_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b \geq 0$ and $\bar{c} = c \; - A^T A_{\mathcal{J}}^{-T} c_{\mathcal{J}} \geq 0$ (no direction of cost decrease). Let $y^\star = -A_{\mathcal{J}}^{-T} c_{\mathcal{J}}$ be a potential solution to the dual problem. Then we have

$$c + A^T y^\star \geq 0$$

(i.e., $y^\star$ is dual feasible) and

$$p^\star = c^T x^\star = c_{\mathcal{J}}^T A_{\mathcal{J}}^{-1} b = -{y^\star}^T b$$

i.e., $y^\star$ is an optimal dual solution, and $d^\star = p^\star$

Duality: general case

General constrained optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, i = 1, \dots, m \\ & h_i(x) = 0, i = 1, \dots, p \end{aligned}$$

Lagrangian given by

$$\mathcal{L}(x, y, z) = f(x) + \sum_{i=1}^{m} z_i g_i(x) + \sum_{i=1}^{p} y_i h_i(x)$$

Analogous to before, we have:

$$\mathcal{L}(x, y, z) = \begin{cases} f(x) & x \text{ feasible} \\ \infty & \text{otherwise} \end{cases}$$

$$p^\star = \min_{x} \max_{y,z \geq 0} \mathcal{L}(x, y, z)$$

KKT conditions

Because optimal $(x^\star, y^\star, z^\star)$ must have $f(x^\star) = \mathcal{L}(x^\star, y^\star, z^\star)$ and $\nabla_x \mathcal{L}(x^\star, y^\star, z^\star) = 0$, we write optimality conditions for the problem as:

1. $g_i(x^\star) \leq 0,\ i = 1, \dots, m$
2. $h_i(x^\star) = 0,\ i = 1, \dots, p$
3. $z_i^\star \geq 0, \mathrm{i} = 1, \dots, m$
4. $\sum_{i=1}^m z_i^\star g_i(x^\star) = 0 \implies z_i^\star g_i(x^\star) = 0, \quad i = 1, \dots, m$
5. $\nabla_x f(x^\star) + \sum_{i=1}^m z_i^\star \nabla_x g_i(x^\star) + \sum_{i=1}^p y_i^\star \nabla_x h_i(x^\star) = 0$

These are called the Karush-Kuhn-Tucker (KKT) equations, and they provide necessary and sufficient conditions for optimal solutions to convex optimization problems

KKT conditions for linear program

For a standard for linear program, the KKT conditions take the form

1. $x \geq 0$
2. $Ax = b$
3. $z \geq 0$
4. $x \circ z = 0$ (where $\circ$ denotes elementwise multiplication)
5. $c + A^T y - z = 0$

A set of *non-linear* equations due to the condition $x \circ z = 0$

Interior point primal dual algorithms (state of the art for solving single large-scale LPs): use Newton's method to find a root of (smoothed version of) these nonlinear equations, smoothing complementarity condition to $x \circ z = t1$ and taking $t \to 0$

Outline

Dual simplex algorithm

In light of discussion on duality, simplex algorithm can be viewed in the following manner:

1. At all steps, maintain primal feasible solution $x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b \geq 0$
2. Work to obtain dual feasible solution $y = -A_{\mathcal{J}}^{-T} c_{\mathcal{J}}$, i.e. with $-A^T y \leq c$

There is an alternative approach, called the dual simplex algorithm, that works in the opposite manner

1. At all steps, maintain dual feasible solution $y = -A_{\mathcal{J}}^{-T} c_{\mathcal{J}}$ such that $-A^T y \leq c$
2. Work to obtain primal feasible solution $x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b \geq 0$

Outline of dual simplex algorithm

Without derivation (it is similar to standard simplex algorithm), dual simplex takes the following form

Given set of indices $\mathcal{J}$ such that $y = -A_{\mathcal{J}}^{-T} c_{\mathcal{J}}$ satisfies $-A^T y \leq c$

Repeat:

1. Letting $x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b$, find some k such that $x_{\mathcal{J}_k} < 0$ (if no index exists, we are done)
2. Let $v = A^T (A_{\mathcal{J}}^{-T})_k$, and determine index to add
$$j^\star = \min_{j \notin \mathcal{J}, v_j < 0} -\bar{c}_i / v_i$$
3. Update index set
$$\mathcal{J} \rightarrow \mathcal{J} - \{\mathcal{J}_k\} \cup \{j^\star\}$$

Incrementally adding constraints

In general, little reason to prefer dual simplex over original simplex

Advantage comes when we can easily find a dual feasible solution, but not a primal feasible solution, arises for instance when we to incrementally add constraints to primal problem

Consider LP in standard form, and suppose we have an optimal primal/dual solution $(x^\star, y^\star)$

Now suppose we want to add constraint $g^T x \leq h$, adding variable x_{n+1} as slack, our new equality constraint in standard form becomes

$$\begin{bmatrix} A & 0 \\ g^T & 1 \end{bmatrix} \begin{bmatrix} x \\ x_{n+1} \end{bmatrix} = \begin{bmatrix} b \\ h \end{bmatrix}$$

Dual feasible solution for added constraint

Not easy to see how we can modify primal solution $x^\star$ to satisfy new equality constraint

But it is trivial to see that $\begin{bmatrix} y^\star \\ 0 \end{bmatrix}$ is a dual feasible solution:

$$\begin{bmatrix} A^T & g \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y^\star \\ 0 \end{bmatrix} \leq \begin{bmatrix} c \\ 0 \end{bmatrix}$$

Thus, we can initialize dual simplex with this starting point ("close" to previous solution), which often takes *much* less time to solve than starting without a good initial solution

This is particularly important in domains like integer programming, where we will modify LPs by adding one additional constraint at a time