15-780 – Graduate Artificial Intelligence: Linear programming

J. Zico Kolter (this lecture) and Ariel Procaccia
Carnegie Mellon University
Spring 2017

Outline

Introduction

Linear programming

Simplex algorithm

Outline

Introduction

Linear programming

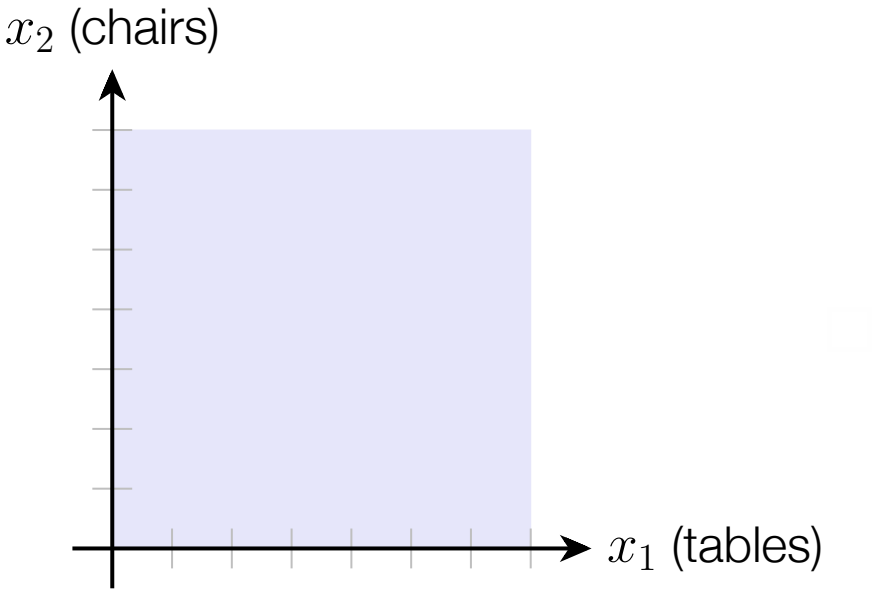
Simplex algorithm

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?

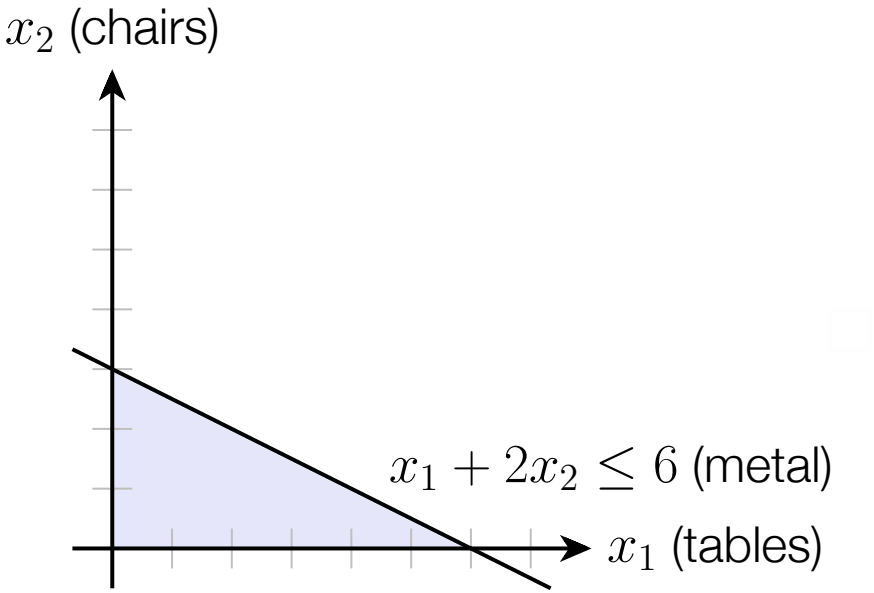
Example: manufacturing

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?



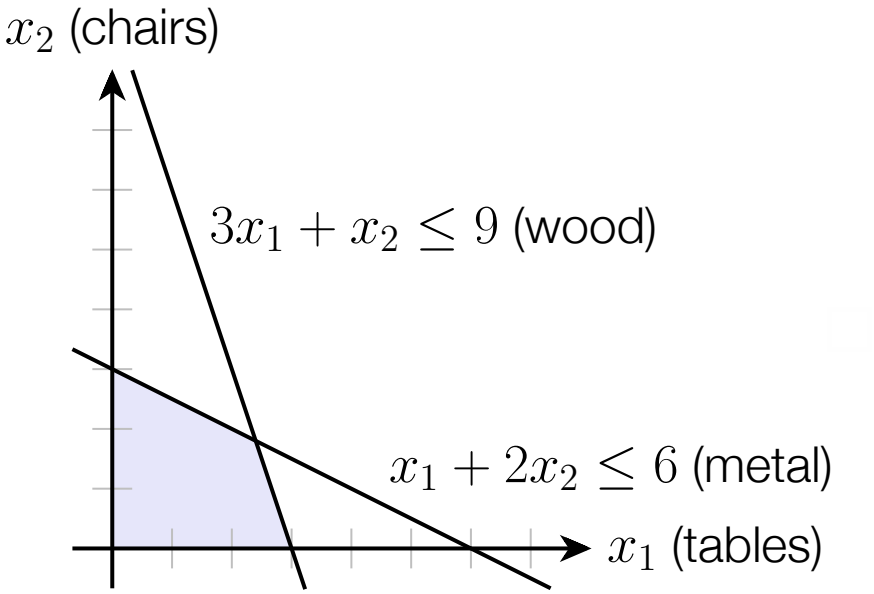
Example: manufacturing

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?



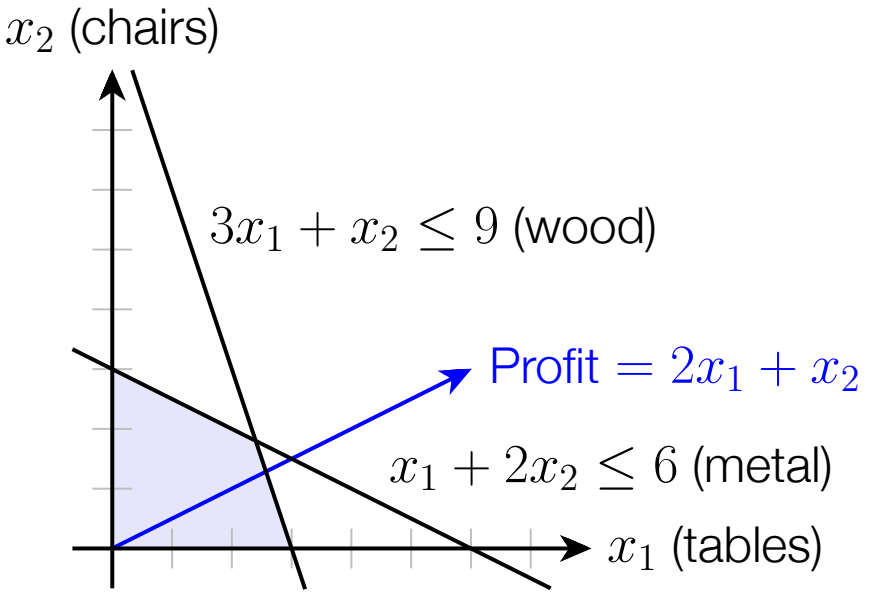
Example: manufacturing

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?



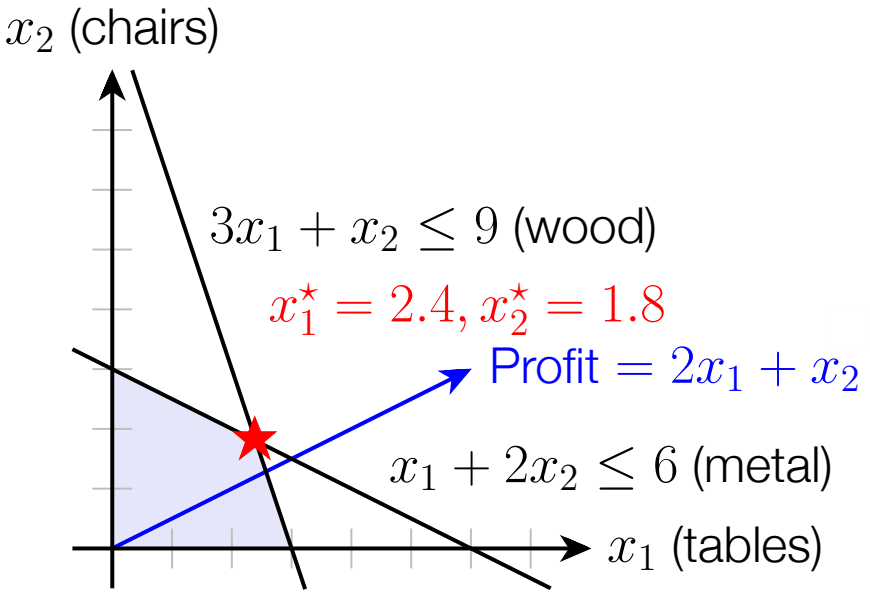
Example: manufacturing

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?



Many applications

Finding optimal strategy in zero-sum two player games

Finding most probable assignment in probabilistic models

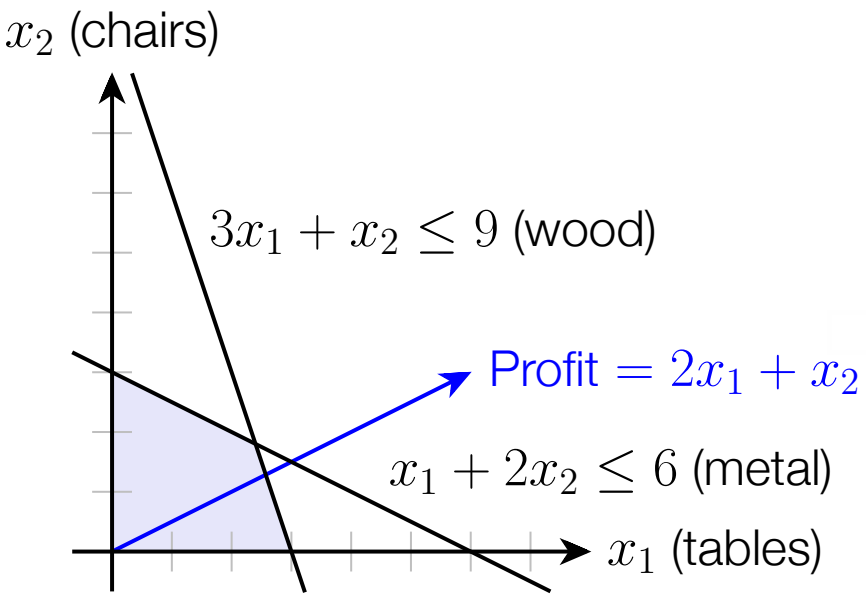
Finding solution in Markov decision processes

Min-cut / max-flow network problems

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

Example manufacturing

We can write our manufacturing problem formally as:



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

Outline

Introduction

Linear programming

Simplex algorithm

Inequality form linear programs

Using linear algebra notation, we can write problems compactly as

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

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:

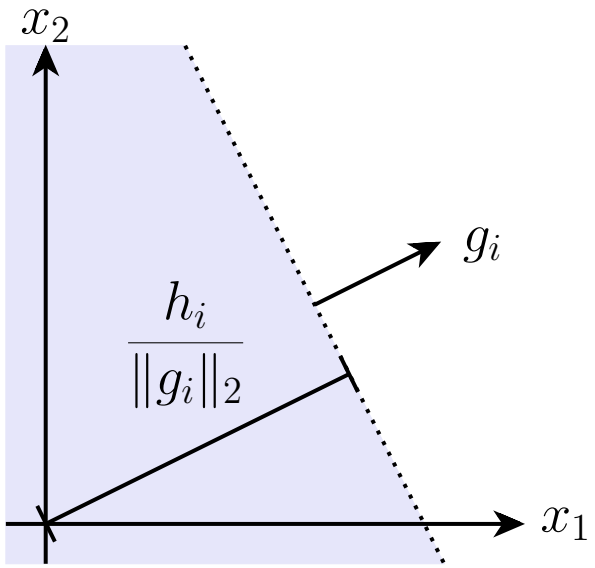
$$\begin{array}{ll} \underset{x_1, x_2}{\text{maximize}} & 2x_1 + x_2 \\ \text{subject to} & 3x_1 + x_2 \leq 9 \\ & x_1 + 2x_2 \leq 6 \\ & x_1, x_2 \geq 0 \end{array} \quad \Longleftrightarrow \quad c = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, G = \begin{bmatrix} 3 & 1 \\ 1 & 2 \\ -1 & 0 \\ 0 & -1 \end{bmatrix},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

Geometry of linear programs

Consider a single row of the constraint matrix

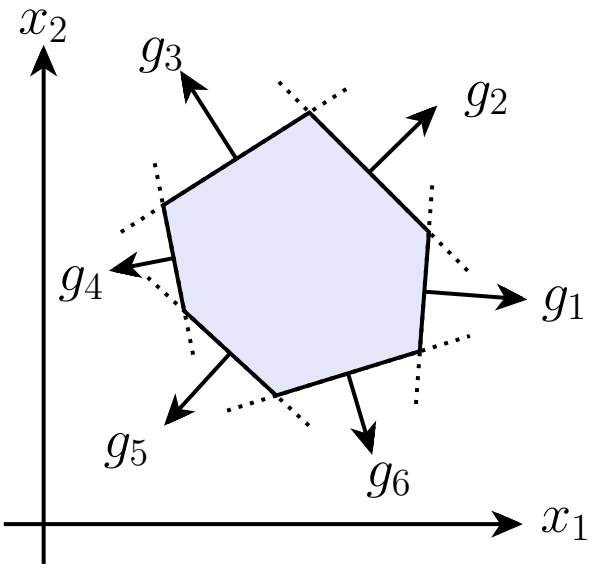
$$g_i^T x \leq h_i$$

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. $O(2^n)$
4. $O(n^n)$

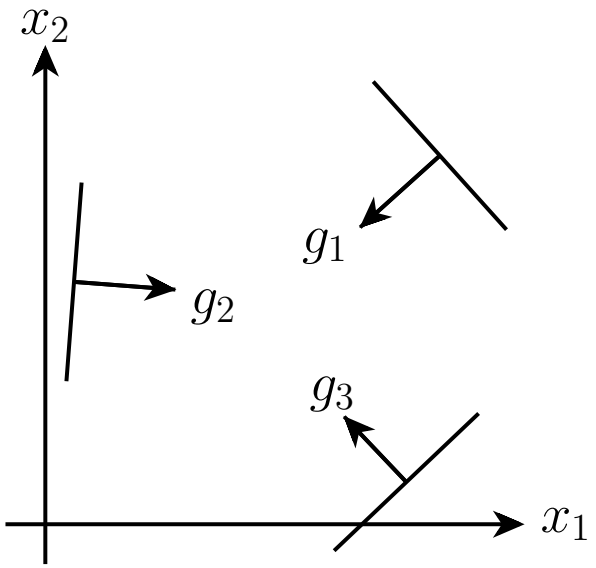
Poll: Number of inequalities in a polytope

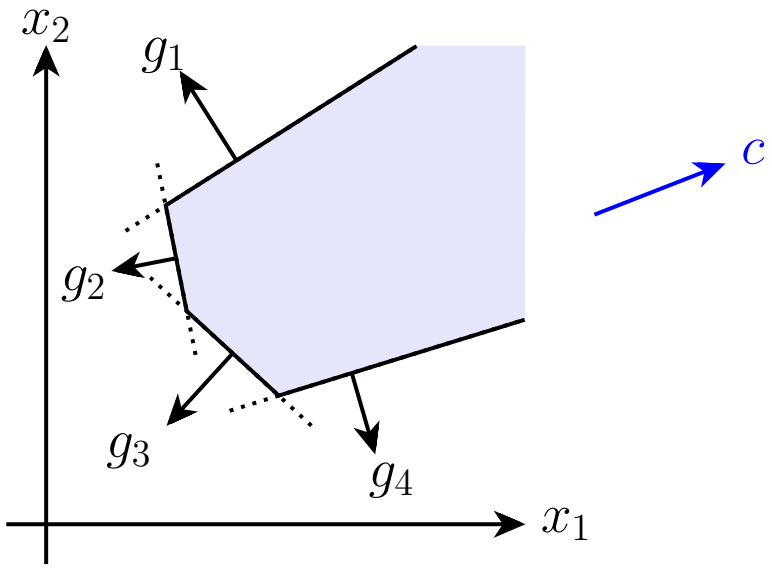
Consider a polytope in n dimensional space, with $O(n)$ corners (vertices). How many linear inequalities could we need to define the polytope?

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





Stanford form linear programs

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

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

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:

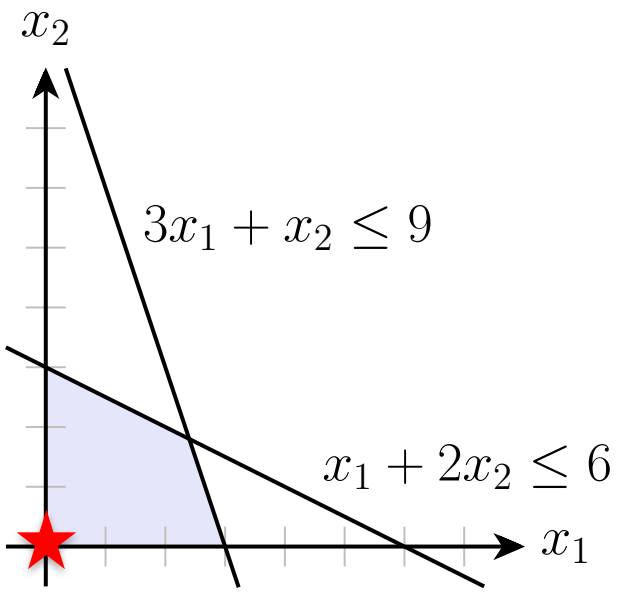
$$A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

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

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

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

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



Finding corners in polytope

Polytope from our example:

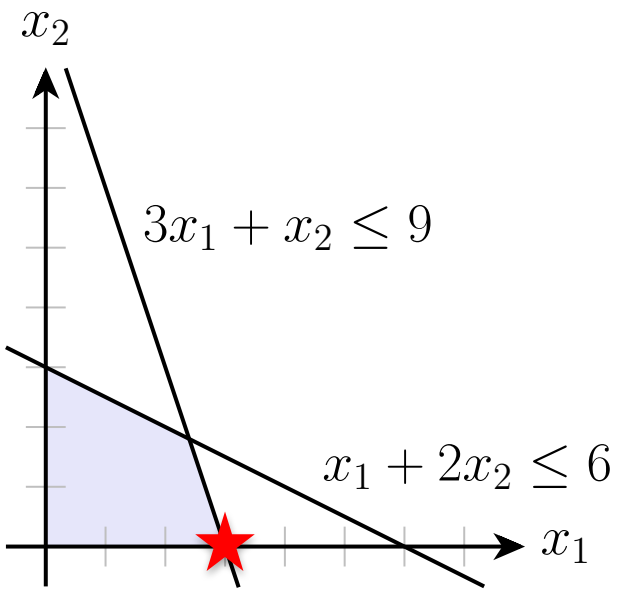
$$A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

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

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

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

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



Finding corners in polytope

Polytope from our example:

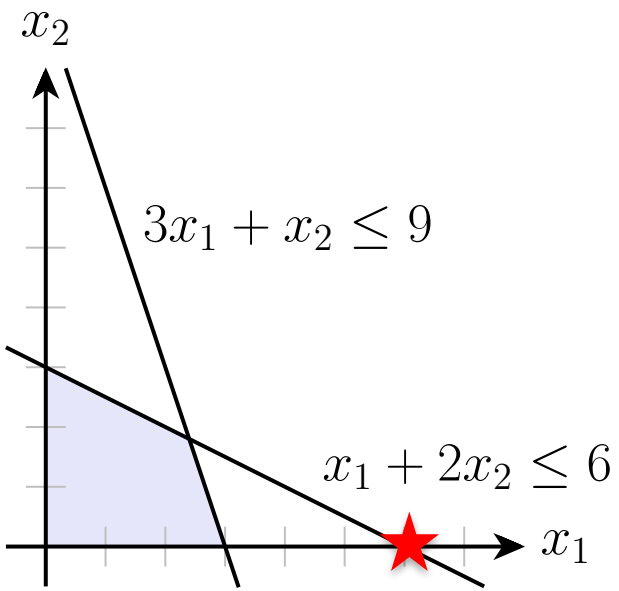
$$A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

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

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

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

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



Outline

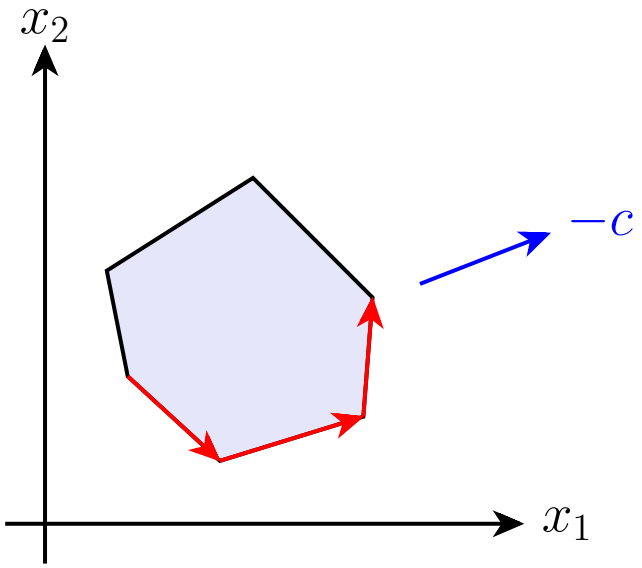
Introduction

Linear programming

Simplex algorithm

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 $j \notin \mathcal{I}$ (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

$$A_{\mathcal{I}} x_{\mathcal{I}} = b \implies A_{\mathcal{I}} x_{\mathcal{I}} + \alpha A_j \neq b$$

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

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

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:

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

or we would leave the feasible set

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

Repeat:

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

2. Find j for which $\bar{c}_j = c_j - c_{\mathcal{I}}^T A_{\mathcal{I}}^{-1} A_j < 0$ (if none exists, return x)

3. Compute step direction $d_{\mathcal{I}} = -A_{\mathcal{I}}^{-1} A_j$ and determine index to remove (or return bounded if $d_{\mathcal{I}} \geq 0$)
$$i^\star = \underset{i \in \mathcal{I}: d_i < 0}{\operatorname{argmin}} -x_i / d_i$$

4. Update index set: $\mathcal{I} \leftarrow \mathcal{I} - \{i^\star\} \cup \{j\}$

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. $O(m^2 + mn)$

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