Linear Programming

Eric Price

UT Austin

CS 331H

Class Outline

Linear Programming

- General way of writing problems: maximize linear function subject to linear constraints.

Linear Programming

- General way of writing problems: maximize linear function subject to linear constraints.
- Developed 1939 by Leonid Kantorovich

Linear Programming

- General way of writing problems: maximize linear function subject to linear constraints.
- Developed 1939 by Leonid Kantorovich
- Think central planning of factories:

Linear Programming

- General way of writing problems: maximize linear function subject to linear constraints.
- Developed 1939 by Leonid Kantorovich
- Think central planning of factories:
  - Can produce cars or trucks

Linear Programming

- General way of writing problems: maximize linear function subject to linear constraints.
- Developed 1939 by Leonid Kantorovich
- Think central planning of factories:
  - Can produce cars or trucks
  - Cars take 2 tons metal, 1 ton wood

Linear Programming

- General way of writing problems: maximize linear function subject to linear constraints.
- Developed 1939 by Leonid Kantorovich
- Think central planning of factories:
  - Can produce cars or trucks
  - Cars take 2 tons metal, 1 ton wood
  - Trucks take 3 tons metal, 5 tons wood

Linear Programming

- General way of writing problems: maximize linear function subject to linear constraints.
- Developed 1939 by Leonid Kantorovich
- Think central planning of factories:
  - Can produce cars or trucks
  - Cars take 2 tons metal, 1 ton wood
  - Trucks take 3 tons metal, 5 tons wood
  - Trucks carry twice as much as cars.

Linear Programming

- General way of writing problems: maximize linear function subject to linear constraints.
- Developed 1939 by Leonid Kantorovich
- Think central planning of factories:
  - Can produce cars or trucks
  - Cars take 2 tons metal, 1 ton wood
  - Trucks take 3 tons metal, 5 tons wood
  - Trucks carry twice as much as cars.
  - You are supplied 12 tons metal, 15 tons wood / day.

Linear Programming

- General way of writing problems: maximize linear function subject to linear constraints.
- Developed 1939 by Leonid Kantorovich
- Think central planning of factories:
  - Can produce cars or trucks
  - Cars take 2 tons metal, 1 ton wood
  - Trucks take 3 tons metal, 5 tons wood
  - Trucks carry twice as much as cars.
  - You are supplied 12 tons metal, 15 tons wood / day.
  - Q: how many cars vs trucks to produce to maximize total capacity?

Linear Programming

- General way of writing problems: maximize linear function subject to linear constraints.
- Developed 1939 by Leonid Kantorovich
- Think central planning of factories:
  - Can produce cars or trucks
  - Cars take 2 tons metal, 1 ton wood
  - Trucks take 3 tons metal, 5 tons wood
  - Trucks carry twice as much as cars.
  - You are supplied 12 tons metal, 15 tons wood / day.
  - Q: how many cars vs trucks to produce to maximize total capacity?
- Mathematically:

$$\begin{aligned} \text{maximize:} \quad & C + 2T \\ \text{subject to:} \quad & 2C + 3T \leq 12 \\ & C + 5T \leq 15 \\ & C, T \geq 0 \end{aligned}$$

Solving small cases by hand

$$\begin{aligned}
\text{maximize:} \quad & C + 2T \\
\text{subject to:} \quad & 2C + 3T \leq 12 \\
& C + 5T \leq 15 \\
& C, T \geq 0
\end{aligned}$$

Solving small cases by hand

$$
\begin{aligned}
\text{maximize:} \quad & C + 2T \\
\text{subject to:} \quad & 2C + 3T \leq 12 \\
& C + 5T \leq 15 \\
& C, T \geq 0
\end{aligned}
$$

- Algebraically:
  - Find all vertices, and for each:
  - Check if feasible (satisfy the constraints)
  - Pick the feasible vertex maximizing the objective.

Solving small cases by hand

$$\begin{aligned} \text{maximize:} \quad & C + 2T \\ \text{subject to:} \quad & 2C + 3T \leq 12 \\ & C + 5T \leq 15 \\ & C, T \geq 0 \end{aligned}$$

- Algebraically:
  - Find all vertices, and for each:
  - Check if feasible (satisfy the constraints)
  - Pick the feasible vertex maximizing the objective.
- Geometrically:
  - Draw the picture of all feasible points
  - Slide in the direction of the objective until you get stuck.

General Linear Programming (LP)

Linear Programming

Optimize (maximize or minimize) a *linear objective* in many variables, subject to *linear constraints* on them $(=, \leq, \geq)$.

General Linear Programming (LP)

Linear Programming

Optimize (maximize or minimize) a *linear objective* in many variables, subject to *linear constraints* on them ($=, \leq, \geq$).

$$
\begin{aligned}
\text{maximize:} \quad & x_1 + 3x_2 - 345x_3 + x_4 \\
\text{subject to:} \quad & x_1 - 17x_2 \leq x_4 + 12 \\
& x_4 - x_3 \geq x_2 \\
& 67x_2 - 3x_1 = 83 \\
& x_3 \leq 0
\end{aligned}
$$

Formulations of LP

Standard form (or "symmetric")

For m constraints on n variables, given $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$:

$$\begin{aligned} \text{maximize:} \quad & c \cdot x \\ \text{subject to:} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

Formulations of LP

Standard form (or "symmetric")

For m constraints on n variables, given $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$:

$$\begin{aligned} \text{maximize:} \quad & c \cdot x \\ \text{subject to:} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

Common alternative forms

"Alternative form"

$$\begin{aligned} \text{maximize:} \quad & c \cdot x \\ \text{subject to:} \quad & Ax \leq b \end{aligned}$$

or

"Slack form"

$$\begin{aligned} \text{maximize:} \quad & c \cdot x \\ \text{subject to:} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

The forms are reducible to each other

Standard	Alternative	Slack
$\max c \cdot x$	$\max c \cdot x$	$\max c \cdot x$
$Ax \leq b$	$Ax \leq b$	$Ax = b$
$x \geq 0$		$x \geq 0$

The forms are reducible to each other

Standard	Alternative	Slack
$\max c \cdot x$	$\max c \cdot x$	$\max c \cdot x$
$Ax \leq b$	$Ax \leq b$	$Ax = b$
$x \geq 0$		$x \geq 0$

- Standard $\rightarrow$ alternative:

The forms are reducible to each other

Standard	Alternative	Slack
$\max c \cdot x$	$\max c \cdot x$	$\max c \cdot x$
$Ax \leq b$	$Ax \leq b$	$Ax = b$
$x \geq 0$		$x \geq 0$

- Standard $\rightarrow$ alternative: solve alternative with $A' = \begin{bmatrix} A \\ -I_n \end{bmatrix}$, $b' = \begin{bmatrix} b \\ 0 \end{bmatrix}$

The forms are reducible to each other

Standard	Alternative	Slack
$\max c \cdot x$	$\max c \cdot x$	$\max c \cdot x$
$Ax \leq b$	$Ax \leq b$	$Ax = b$
$x \geq 0$		$x \geq 0$

- Standard $\rightarrow$ alternative: solve alternative with $A' = \begin{bmatrix} A \\ -I_n \end{bmatrix}$, $b' = \begin{bmatrix} b \\ 0 \end{bmatrix}$
- Alternative $\rightarrow$ standard:

The forms are reducible to each other

Standard	Alternative	Slack
$\max c \cdot x$	$\max c \cdot x$	$\max c \cdot x$
$Ax \leq b$	$Ax \leq b$	$Ax = b$
$x \geq 0$		$x \geq 0$

- Standard $\rightarrow$ alternative: solve alternative with $A' = \begin{bmatrix} A \\ -I_n \end{bmatrix}$, $b' = \begin{bmatrix} b \\ 0 \end{bmatrix}$
- Alternative $\rightarrow$ standard: new nonnegative variables y and z, so $x = y - z$. Solve standard with $A' = \begin{bmatrix} A & -A \end{bmatrix}$.

The forms are reducible to each other

Standard	Alternative	Slack
$\max c \cdot x$	$\max c \cdot x$	$\max c \cdot x$
$Ax \leq b$	$Ax \leq b$	$Ax = b$
$x \geq 0$		$x \geq 0$

- Standard $\rightarrow$ alternative: solve alternative with $A' = \begin{bmatrix} A \\ -I_n \end{bmatrix}$, $b' = \begin{bmatrix} b \\ 0 \end{bmatrix}$
- Alternative $\rightarrow$ standard: new nonnegative variables y and z, so $x = y - z$. Solve standard with $A' = \begin{bmatrix} A & -A \end{bmatrix}$.
- Slack $\rightarrow$ standard:

The forms are reducible to each other

Standard	Alternative	Slack
$\max c \cdot x$	$\max c \cdot x$	$\max c \cdot x$
$Ax \leq b$	$Ax \leq b$	$Ax = b$
$x \geq 0$		$x \geq 0$

- Standard $\rightarrow$ alternative: solve alternative with $A' = \begin{bmatrix} A \\ -I_n \end{bmatrix}$, $b' = \begin{bmatrix} b \\ 0 \end{bmatrix}$
- Alternative $\rightarrow$ standard: new nonnegative variables y and z, so $x = y - z$. Solve standard with $A' = \begin{bmatrix} A & -A \end{bmatrix}$.
- Slack $\rightarrow$ standard: solve standard with $A' = \begin{bmatrix} A \\ -A \end{bmatrix}$, $b' = \begin{bmatrix} b \\ -b \end{bmatrix}$.

The forms are reducible to each other

Standard	Alternative	Slack
$\max c \cdot x$	$\max c \cdot x$	$\max c \cdot x$
$Ax \leq b$	$Ax \leq b$	$Ax = b$
$x \geq 0$		$x \geq 0$

- Standard $\to$ alternative: solve alternative with $A' = \begin{bmatrix} A \\ -I_n \end{bmatrix}$, $b' = \begin{bmatrix} b \\ 0 \end{bmatrix}$
- Alternative $\to$ standard: new nonnegative variables y and z, so $x = y - z$. Solve standard with $A' = \begin{bmatrix} A & -A \end{bmatrix}$.
- Slack $\to$ standard: solve standard with $A' = \begin{bmatrix} A \\ -A \end{bmatrix}$, $b' = \begin{bmatrix} b \\ -b \end{bmatrix}$.
- Standard $\to$ slack:

The forms are reducible to each other

Standard	Alternative	Slack
$\max c \cdot x$	$\max c \cdot x$	$\max c \cdot x$
$Ax \leq b$	$Ax \leq b$	$Ax = b$
$x \geq 0$		$x \geq 0$

- Standard $\to$ alternative: solve alternative with $A' = \begin{bmatrix} A \\ -I_n \end{bmatrix}$, $b' = \begin{bmatrix} b \\ 0 \end{bmatrix}$
- Alternative $\to$ standard: new nonnegative variables y and z, so $x = y - z$. Solve standard with $A' = \begin{bmatrix} A & -A \end{bmatrix}$.
- Slack $\to$ standard: solve standard with $A' = \begin{bmatrix} A \\ -A \end{bmatrix}$, $b' = \begin{bmatrix} b \\ -b \end{bmatrix}$.
- Standard $\to$ slack: m new "slack" variables z, solve slack with $A' = \begin{bmatrix} A & I_m \end{bmatrix}$

The forms are reducible to each other

Standard	Alternative	Slack
$\max c \cdot x$	$\max c \cdot x$	$\max c \cdot x$
$Ax \leq b$	$Ax \leq b$	$Ax = b$
$x \geq 0$		$x \geq 0$

- Standard $\to$ alternative: solve alternative with $A' = \begin{bmatrix} A \\ -I_n \end{bmatrix}$, $b' = \begin{bmatrix} b \\ 0 \end{bmatrix}$
- Alternative $\to$ standard: new nonnegative variables y and z, so $x = y - z$. Solve standard with $A' = \begin{bmatrix} A & -A \end{bmatrix}$.
- Slack $\to$ standard: solve standard with $A' = \begin{bmatrix} A \\ -A \end{bmatrix}$, $b' = \begin{bmatrix} b \\ -b \end{bmatrix}$.
- Standard $\to$ slack: m new "slack" variables z, solve slack with $A' = \begin{bmatrix} A & I_m \end{bmatrix}$
- Minimization problems?

The forms are reducible to each other

Standard	Alternative	Slack
$\max c \cdot x$	$\max c \cdot x$	$\max c \cdot x$
$Ax \leq b$	$Ax \leq b$	$Ax = b$
$x \geq 0$		$x \geq 0$

- Standard $\to$ alternative: solve alternative with $A' = \begin{bmatrix} A \\ -I_n \end{bmatrix}$, $b' = \begin{bmatrix} b \\ 0 \end{bmatrix}$
- Alternative $\to$ standard: new nonnegative variables y and z, so $x = y - z$. Solve standard with $A' = \begin{bmatrix} A & -A \end{bmatrix}$.
- Slack $\to$ standard: solve standard with $A' = \begin{bmatrix} A \\ -A \end{bmatrix}$, $b' = \begin{bmatrix} b \\ -b \end{bmatrix}$.
- Standard $\to$ slack: m new "slack" variables z, solve slack with $A' = \begin{bmatrix} A & I_m \end{bmatrix}$
- Minimization problems? $c' = -c$.

Class Outline

Overview of solution methods

- Simplex

Overview of solution methods

- Simplex
- Ellipsoid

Overview of solution methods

- Simplex
- Ellipsoid
- Interior point

Overview of solution methods

- Simplex
  - Start at a vertex, and walk from vertex to vertex, increasing objective.
- Ellipsoid
- Interior point

Overview of solution methods

- Simplex
  - Start at a vertex, and walk from vertex to vertex, increasing objective.
  - First algorithm (Dantzig '47)
- Ellipsoid
- Interior point

Overview of solution methods

- Simplex
  - Start at a vertex, and walk from vertex to vertex, increasing objective.
  - First algorithm (Dantzig '47)
  - Worst-case inputs can take exponential time, but fast on *most* inputs.
- Ellipsoid
- Interior point

Overview of solution methods

- Simplex
  - Start at a vertex, and walk from vertex to vertex, increasing objective.
  - First algorithm (Dantzig '47)
  - Worst-case inputs can take exponential time, but fast on *most* inputs.
- Ellipsoid
  - Iteratively shrink an ellipsoid around the solution.
- Interior point

Overview of solution methods

- Simplex
  - Start at a vertex, and walk from vertex to vertex, increasing objective.
  - First algorithm (Dantzig '47)
  - Worst-case inputs can take exponential time, but fast on *most* inputs.
- Ellipsoid
  - Iteratively shrink an ellipsoid around the solution.
  - First polynomial time algorithm (Khachiyan '79); $O(n^6 L)$ for L bits of precision.
- Interior point

Overview of solution methods

- Simplex
  - Start at a vertex, and walk from vertex to vertex, increasing objective.
  - First algorithm (Dantzig '47)
  - Worst-case inputs can take exponential time, but fast on *most* inputs.
- Ellipsoid
  - Iteratively shrink an ellipsoid around the solution.
  - First polynomial time algorithm (Khachiyan '79); $O(n^6 L)$ for L bits of precision.
- Interior point
  - Iteratively move through the center of the region.

Overview of solution methods

- Simplex
  - Start at a vertex, and walk from vertex to vertex, increasing objective.
  - First algorithm (Dantzig '47)
  - Worst-case inputs can take exponential time, but fast on *most* inputs.
- Ellipsoid
  - Iteratively shrink an ellipsoid around the solution.
  - First polynomial time algorithm (Khachiyan '79); $O(n^6 L)$ for L bits of precision.
- Interior point
  - Iteratively move through the center of the region.
  - Introduced by Karmarkar in '84, $O(n^{3.5})$.

Overview of solution methods

- Simplex
  - Start at a vertex, and walk from vertex to vertex, increasing objective.
  - First algorithm (Dantzig '47)
  - Worst-case inputs can take exponential time, but fast on *most* inputs.
- Ellipsoid
  - Iteratively shrink an ellipsoid around the solution.
  - First polynomial time algorithm (Khachiyan '79); $O(n^6 L)$ for L bits of precision.
- Interior point
  - Iteratively move through the center of the region.
  - Introduced by Karmarkar in '84, $O(n^{3.5})$.
  - More practical than ellipsoid, even better than simplex sometimes.

Overview of solution methods

- Simplex
  - Start at a vertex, and walk from vertex to vertex, increasing objective.
  - First algorithm (Dantzig '47)
  - Worst-case inputs can take exponential time, but fast on *most* inputs.
- Ellipsoid
  - Iteratively shrink an ellipsoid around the solution.
  - First polynomial time algorithm (Khachiyan '79); $O(n^6 L)$ for L bits of precision.
- Interior point
  - Iteratively move through the center of the region.
  - Introduced by Karmarkar in '84, $O(n^{3.5})$.
  - More practical than ellipsoid, even better than simplex sometimes.
  - Best theoretical result: $O(n^{2.38} L)$ time (Cohen, Lee, Song '19).

Simplex Algorithm

- Linear program with n variables and m constraints (including $x_i \geq 0$).

Simplex Algorithm

- Linear program with n variables and m constraints (including $x_i \geq 0$).
- Vertex of feasible set is where some n constraints are tight.

Simplex Algorithm

- Linear program with n variables and m constraints (including $x_i \geq 0$).
- Vertex of feasible set is where some n constraints are tight.
- n adjacent vertices: drop one constraint, move along line until another constraint becomes tight.

Simplex Algorithm

- Linear program with n variables and m constraints (including $x_i \geq 0$).
- Vertex of feasible set is where some n constraints are tight.
- n adjacent vertices: drop one constraint, move along line until another constraint becomes tight.

Simplex algorithm

Simplex Algorithm

- Linear program with n variables and m constraints (including $x_i \geq 0$).
- Vertex of feasible set is where some n constraints are tight.
- n adjacent vertices: drop one constraint, move along line until another constraint becomes tight.

Simplex algorithm

1. Find an initial vertex (we'll see how later)

Simplex Algorithm

- Linear program with n variables and m constraints (including $x_i \geq 0$).
- Vertex of feasible set is where some n constraints are tight.
- n adjacent vertices: drop one constraint, move along line until another constraint becomes tight.

Simplex algorithm

1. Find an initial vertex (we'll see how later)
2. Repeatedly move to adjacent vertex of larger objective.

Simplex Algorithm

- Linear program with n variables and m constraints (including $x_i \geq 0$).
- Vertex of feasible set is where some n constraints are tight.
- n adjacent vertices: drop one constraint, move along line until another constraint becomes tight.

Simplex algorithm

1. Find an initial vertex (we'll see how later)
2. Repeatedly move to adjacent vertex of larger objective.

- Correctness:

Simplex Algorithm

- Linear program with n variables and m constraints (including $x_i \geq 0$).
- Vertex of feasible set is where some n constraints are tight.
- n adjacent vertices: drop one constraint, move along line until another constraint becomes tight.

Simplex algorithm

1. Find an initial vertex (we'll see how later)
2. Repeatedly move to adjacent vertex of larger objective.

- Correctness:
  - If we get to the true solution, the algorithm will stop.

Simplex Algorithm

- Linear program with n variables and m constraints (including $x_i \geq 0$).
- Vertex of feasible set is where some n constraints are tight.
- n adjacent vertices: drop one constraint, move along line until another constraint becomes tight.

Simplex algorithm

1. Find an initial vertex (we'll see how later)
2. Repeatedly move to adjacent vertex of larger objective.

- Correctness:
  - If we get to the true solution, the algorithm will stop.
  - By convexity: if *not* at the true solution, can move and make progress.

Simplex Algorithm

- Linear program with n variables and m constraints (including $x_i \geq 0$).
- Vertex of feasible set is where some n constraints are tight.
- n adjacent vertices: drop one constraint, move along line until another constraint becomes tight.

Simplex algorithm

1. Find an initial vertex (we'll see how later)
2. Repeatedly move to adjacent vertex of larger objective.

- Correctness:
  - If we get to the true solution, the algorithm will stop.
  - By convexity: if *not* at the true solution, can move and make progress.
- Running time:

Simplex Algorithm

- Linear program with n variables and m constraints (including $x_i \geq 0$).
- Vertex of feasible set is where some n constraints are tight.
- n adjacent vertices: drop one constraint, move along line until another constraint becomes tight.

Simplex algorithm

1. Find an initial vertex (we'll see how later)
2. Repeatedly move to adjacent vertex of larger objective.

- Correctness:
  - If we get to the true solution, the algorithm will stop.
  - By convexity: if *not* at the true solution, can move and make progress.
- Running time:
  - Polynomial time per iteration.

Simplex Algorithm

- Linear program with n variables and m constraints (including $x_i \geq 0$).
- Vertex of feasible set is where some n constraints are tight.
- n adjacent vertices: drop one constraint, move along line until another constraint becomes tight.

Simplex algorithm

1. Find an initial vertex (we'll see how later)
2. Repeatedly move to adjacent vertex of larger objective.

- Correctness:
  - If we get to the true solution, the algorithm will stop.
  - By convexity: if *not* at the true solution, can move and make progress.
- Running time:
  - Polynomial time per iteration.
  - Number of iterations depends on problem instance & rule for choosing next vertex, but could be exponential.

Finding an initial feasible vertex

- Simplex works, eventually, once you have a feasible vertex.
- Doesn't seem so useful:

Problem

If you can solve "does this polytope have any feasible point" you can also solve linear programming (= optimize over polytopes).

Finding an initial feasible vertex

- Simplex works, eventually, once you have a feasible vertex.
- Doesn't seem so useful:

Problem

If you can solve "does this polytope have any feasible point" you can also solve linear programming (= optimize over polytopes).

Proof.

We want to determine $OPT = \max c \cdot x$ s.t. $Ax \leq b$. Then $OPT \geq \tau$ if and only if the polytope

$$Ax \leq b$$
$$c \cdot x \geq \tau$$

has any solution. So if we can solve this, we binary search on τ to solve LP. □

Finding an initial feasible vertex

- Simplex works, eventually, once you have a feasible vertex.

Finding an initial feasible vertex

- Simplex works, eventually, once you have a feasible vertex.
- In general, finding a feasible vertex is as hard as LP.

Finding an initial feasible vertex

- Simplex works, eventually, once you have a feasible vertex.
- In general, finding a feasible vertex is as hard as LP.
- We create a new LP, where finding a feasible vertex is easy, and the optimal solution identifies a feasible point of the initial LP.

Finding an initial feasible vertex

- Simplex works, eventually, once you have a feasible vertex.
- In general, finding a feasible vertex is as hard as LP.
- We create a new LP, where finding a feasible vertex is easy, and the optimal solution identifies a feasible point of the initial LP.

Finding a feasible point

We want to find a point x such that $Ax \leq b, x \geq 0$. Introduce a new variable $z \in \mathbb{R}$, and solve:

$$\begin{aligned} \text{minimize:} \quad & z \\ \text{subject to:} \quad & Ax - z \leq b \\ & x, z \geq 0 \end{aligned} \tag{NEW}$$

Finding an initial feasible vertex

- Simplex works, eventually, once you have a feasible vertex.
- In general, finding a feasible vertex is as hard as LP.
- We create a new LP, where finding a feasible vertex is easy, and the optimal solution identifies a feasible point of the initial LP.

Finding a feasible point

We want to find a point x such that $Ax \leq b, x \geq 0$. Introduce a new variable $z \in \mathbb{R}$, and solve:

$$\begin{aligned} \text{minimize:} \quad & z \\ \text{subject to:} \quad & Ax - z \leq b \\ & x, z \geq 0 \end{aligned} \tag{NEW}$$

- $z = 0$ possible if and only if $Ax \leq b, x \geq 0$ is feasible.

Finding an initial feasible vertex

- Simplex works, eventually, once you have a feasible vertex.
- In general, finding a feasible vertex is as hard as LP.
- We create a new LP, where finding a feasible vertex is easy, and the optimal solution identifies a feasible point of the initial LP.

Finding a feasible point

We want to find a point x such that $Ax \leq b, x \geq 0$. Introduce a new variable $z \in \mathbb{R}$, and solve:

$$\begin{aligned} \text{minimize:} \quad & z \\ \text{subject to:} \quad & Ax - z \leq b \\ & x, z \geq 0 \end{aligned} \tag{NEW}$$

- $z = 0$ possible if and only if $Ax \leq b, x \geq 0$ is feasible.
- $x = 0$, $z = \max(0, b_1, \dots, b_m)$ is a *feasible vertex*.

Finding an initial feasible vertex

- Simplex works, eventually, once you have a feasible vertex.
- In general, finding a feasible vertex is as hard as LP.
- We create a new LP, where finding a feasible vertex is easy, and the optimal solution identifies a feasible point of the initial LP.

Finding a feasible point

We want to find a point x such that $Ax \leq b, x \geq 0$. Introduce a new variable $z \in \mathbb{R}$, and solve:

$$\begin{aligned} \text{minimize:} \quad & z \\ \text{subject to:} \quad & Ax - z \leq b \\ & x, z \geq 0 \end{aligned} \tag{NEW}$$

- $z = 0$ possible if and only if $Ax \leq b, x \geq 0$ is feasible.
- $x = 0$, $z = \max(0, b_1, \ldots, b_m)$ is a *feasible vertex*.
- So simplex can get started on NEW and solve it.

Finding an initial feasible vertex

- Simplex works, eventually, once you have a feasible vertex.

Finding a feasible point

We want to find a point x such that $Ax \leq b, x \geq 0$. Introduce a new variable $z \in \mathbb{R}$, and solve:

$$\begin{aligned} \text{minimize:} \quad & z \\ \text{subject to:} \quad & Ax - z \leq b \\ & x, z \geq 0 \end{aligned} \tag{NEW}$$

- Simplex can get started on NEW and solve it.

Finding an initial feasible vertex

- Simplex works, eventually, once you have a feasible vertex.

Finding a feasible point

We want to find a point x such that $Ax \leq b, x \geq 0$. Introduce a new variable $z \in \mathbb{R}$, and solve:

$$\begin{aligned} \text{minimize:} \quad & z \\ \text{subject to:} \quad & Ax - z \leq b \\ & x, z \geq 0 \end{aligned} \tag{NEW}$$

- Simplex can get started on NEW and solve it.
- The solution to NEW returned by simplex is a vertex $(\widehat{x}, \widehat{z})$ of NEW.

Finding an initial feasible vertex

- Simplex works, eventually, once you have a feasible vertex.

Finding a feasible point

We want to find a point x such that $Ax \leq b, x \geq 0$. Introduce a new variable $z \in \mathbb{R}$, and solve:

$$\begin{aligned} \text{minimize:} \quad & z \\ \text{subject to:} \quad & Ax - z \leq b \\ & x, z \geq 0 \end{aligned} \tag{NEW}$$

- Simplex can get started on NEW and solve it.
- The solution to NEW returned by simplex is a vertex $(\widehat{x}, \widehat{z})$ of NEW.
- NEW has $n+1$ variables, one tight constraint of the optimum is $z \geq 0$, and the other n are among $Ax \leq b, x \geq 0$.

Finding an initial feasible vertex

- Simplex works, eventually, once you have a feasible vertex.

Finding a feasible point

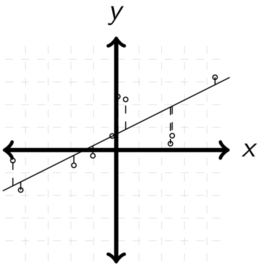
We want to find a point x such that $Ax \leq b, x \geq 0$. Introduce a new variable $z \in \mathbb{R}$, and solve:

$$\begin{aligned} \text{minimize:} \quad & z \\ \text{subject to:} \quad & Ax - z \leq b \\ & x, z \geq 0 \end{aligned} \qquad \text{(NEW)}$$

- Simplex can get started on NEW and solve it.
- The solution to NEW returned by simplex is a vertex $(\widehat{x}, \widehat{z})$ of NEW.
- NEW has $n + 1$ variables, one tight constraint of the optimum is $z \geq 0$, and the other n are among $Ax \leq b, x \geq 0$.
- Hence the solution $\widehat{x}$ is a vertex of the original LP.

Class Outline

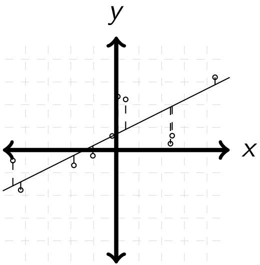
L1 linear regression



Given n points on plane: $(x_1, y_1), \ldots, (x_n, y_n)$. Find the line $mx + b$ minimizing the average error:

$$\text{Err} = \frac{1}{n} \sum_{i=1}^{n} |y_i - (mx_i + b)|$$

L1 linear regression



Given n points on plane: $(x_1, y_1), \ldots, (x_n, y_n)$. Find the line $mx + b$ minimizing the average error:

$$\text{Err} = \frac{1}{n} \sum_{i=1}^{n} |y_i - (mx_i + b)|$$

Part (2): Now, minimize the *maximum* error.

Writing old problems as linear programs

- Write network flow as a linear program
- Write shortest paths as a linear program
- Write minimum cut as a linear program