Convex Optimization

AA 203 Recitation #4

May 1st, 2020

Agenda

Preliminaries

- Why study Convex Optimization?
- Convex Sets & Convex Functions
- Convex Programming
- Linear Matrix Inequalities

Agenda

Preliminaries

- Why study Convex Optimization?
- Convex Sets & Convex Functions
- Convex Programming
- Linear Matrix Inequalities

Optimization Models and Tools

- Solvers (i.e. CPLEX, CVX).
- Linear Programming
- Quadratic Programming

Agenda

Preliminaries

- Why study Convex Optimization?
- Convex Sets & Convex Functions
- Convex Programming
- Linear Matrix Inequalities

Optimization Models and Tools

- Solvers (i.e. CPLEX, CVX).
- Linear Programming
- Quadratic Programming

Algorithms

- Simplex Method
- Cutting Plane Methods (Ellipsoid Method)
- Interior Point Method

Preliminaries

Why study Convex Optimization?

Observation 1: Iterative methods like Gradient method and Newton Method can find local minima.

Why study Convex Optimization?

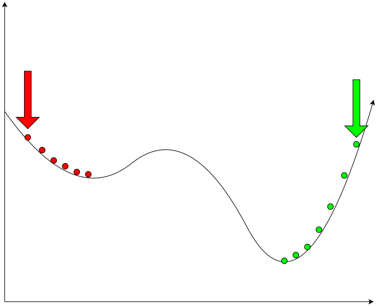
Observation 1: Iterative methods like Gradient method and Newton Method can find local minima.

Observation 2: These methods can also get trapped in local minima and thus fail to converge to the *global* minima.

Why study Convex Optimization?

Observation 1: Iterative methods like Gradient method and Newton Method can find local minima.

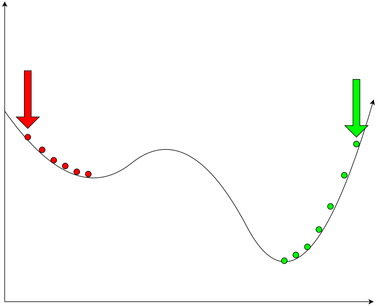
Observation 2: These methods can also get trapped in local minima and thus fail to converge to the *global* minima.



Why study Convex Optimization?

Observation 1: Iterative methods like Gradient method and Newton Method can find local minima.

Observation 2: These methods can also get trapped in local minima and thus fail to converge to the *global* minima.



Observation 3: This issue doesn't show up for convex problems. For convex optimization problems, every locally optimal solution is also globally optimal.

Convex Sets

Definition (Convex Set)

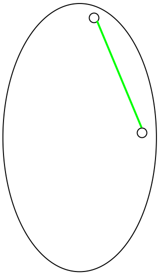
A set $S \subset \mathbb{R}^d$ is convex if and only if: for any $x, y \in S$ and any $\alpha \in [0, 1]$, we also have $\alpha x + (1 - \alpha)y \in S$.

Convex Sets

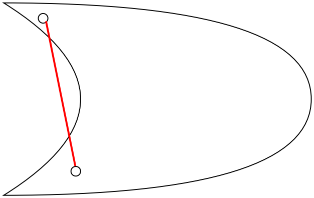
Definition (Convex Set)

A set $S \subset \mathbb{R}^d$ is convex if and only if: for any $x, y \in S$ and any $\alpha \in [0, 1]$, we also have $\alpha x + (1 - \alpha) y \in S$.

Examples:



Yes!



No

Convex Functions

Definition (Convex Functions)

A function $f : S \to \mathbb{R}$ over a convex set $S \subset \mathbb{R}^d$ is convex if the set

$$\text{epigraph}(f) := \left\{(x, y) \in \mathbb{R}^{d+1} : x \in S, y \in \mathbb{R} \text{ and } y \geq f(x)\right\} \text{ is convex.}$$

Convex Functions

Definition (Convex Functions)

A function $f : S \to \mathbb{R}$ over a convex set $S \subset \mathbb{R}^d$ is convex if the set

$$\text{epigraph}(f) := \left\{ (x, y) \in \mathbb{R}^{d+1} : x \in S, y \in \mathbb{R} \text{ and } y \geq f(x) \right\} \text{ is convex.}$$

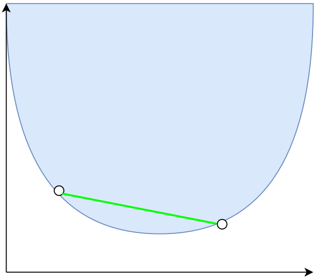
Equivalently: If the chord between $f(x_1)$ and $f(x_2)$ overestimates f between x_1 and x_2.

Definition (Convex Functions)

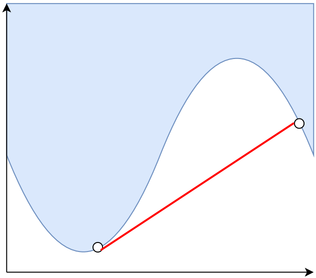
A function $f : S \to \mathbb{R}$ over a convex set $S \subset \mathbb{R}^d$ is convex if the set

$$\text{epigraph}(f) := \left\{(x, y) \in \mathbb{R}^{d+1} : x \in S, y \in \mathbb{R} \text{ and } y \geq f(x)\right\} \text{ is convex.}$$

Equivalently: If the chord between $f(x_1)$ and $f(x_2)$ overestimates f between x_1 and x_2. Examples:



Yes!



No

Convex Program

Definition (Convex Program)

A convex program (aka convex optimization problem) is a minimization problem of a convex function over a convex set:

$$\text{minimize } f(x)$$
$$\text{subject to } x \in S$$

where S is a convex set and $f : S \to \mathbb{R}$ is a convex function.

Convex Program

Definition (Convex Program)

A convex program (aka convex optimization problem) is a minimization problem of a convex function over a convex set:

$$\text{minimize } f(x)$$
$$\text{subject to } x \in S$$

where S is a convex set and $f : S \to \mathbb{R}$ is a convex function.

Definition (Local Minimum)

For an optimization problem $\min_{x \in S} f(x)$, a point x^* is a local minimum if there exists some $\epsilon > 0$ so that for every $x \in S$ with $||x - x^*||_2 \leq \epsilon$, $f(x^*) \leq f(x)$.

Convex Program: Local Optima are Global Optima

Theorem (Equivalence of Local and Global Optima)

Let $\min_{x \in S} f(x)$ *be a convex program. If* x^* *is a local minimum, then* $f(x^*) \leq f(x)$ *for every* $x \in S$*. In other words,* x^* *is a global minimum.*

Convex Program: Local Optima are Global Optima

Theorem (Equivalence of Local and Global Optima)

Let $\min_{x \in S} f(x)$ *be a convex program. If* x^* *is a local minimum, then* $f(x^*) \leq f(x)$ *for every* $x \in S$*. In other words,* x^* *is a global minimum.*

Proof: (by contradiction) Suppose x^* is a local but not global minimum.

Convex Program: Local Optima are Global Optima

Theorem (Equivalence of Local and Global Optima)

Let $\min_{x \in S} f(x)$ *be a convex program. If* x^* *is a local minimum, then* $f(x^*) \leq f(x)$ *for every* $x \in S$*. In other words,* x^* *is a global minimum.*

Proof: (by contradiction) Suppose x^* is a local but not global minimum.

Since x^* is a local optima, there exists $\epsilon > 0$ so that $f(x^*) \leq f(x)$ for all $x \in S$, $||x - x^*||_2 \leq \epsilon$.

Convex Program: Local Optima are Global Optima

Theorem (Equivalence of Local and Global Optima)

Let $\min_{x \in S} f(x)$ *be a convex program. If* x^* *is a local minimum, then* $f(x^*) \leq f(x)$ *for every* $x \in S$*. In other words,* x^* *is a global minimum.*

Proof: (by contradiction) Suppose x^* is a local but not global minimum.

Since x^* is a local optima, there exists $\epsilon > 0$ so that $f(x^*) \leq f(x)$ for all $x \in S$, $||x - x^*||_2 \leq \epsilon$.

Since x^* is not a global minimum, we can find $x_0 \in S$ where $f(x_0) < f(x^*)$.

Convex Program: Local Optima are Global Optima

Theorem (Equivalence of Local and Global Optima)

Let $\min_{x \in S} f(x)$ *be a convex program. If* x^* *is a local minimum, then* $f(x^*) \leq f(x)$ *for every* $x \in S$*. In other words,* x^* *is a global minimum.*

Proof: (by contradiction) Suppose x^* is a local but not global minimum.

- Since x^* is a local optima, there exists $\epsilon > 0$ so that $f(x^*) \leq f(x)$ for all $x \in S$, $||x - x^*||_2 \leq \epsilon$.
- Since x^* is not a global minimum, we can find $x_0 \in S$ where $f(x_0) < f(x^*)$.
- Since S is convex, $\alpha x^* + (1 - \alpha)x_0 \in S$ for every $\alpha \in [0, 1]$.

Convex Program: Local Optima are Global Optima

Theorem (Equivalence of Local and Global Optima)

Let $\min_{x \in S} f(x)$ be a convex program. If x^* is a local minimum, then $f(x^*) \leq f(x)$ for every $x \in S$. In other words, x^* is a global minimum.

Proof: (by contradiction) Suppose x^* is a local but not global minimum.

Since x^* is a local optima, there exists $\epsilon > 0$ so that $f(x^*) \leq f(x)$ for all $x \in S$, $||x - x^*||_2 \leq \epsilon$.

Since x^* is not a global minimum, we can find $x_0 \in S$ where $f(x_0) < f(x^*)$.

Since S is convex, $\alpha x^* + (1-\alpha)x_0 \in S$ for every $\alpha \in [0,1]$.

Note that $f((1-\alpha)x^* + \alpha x_0) \leq (1-\alpha)f(x^*) + \alpha f(x_0) < f(x^*)$.

Convex Program: Local Optima are Global Optima

Theorem (Equivalence of Local and Global Optima)

Let $\min_{x \in S} f(x)$ *be a convex program. If* x^* *is a local minimum, then* $f(x^*) \leq f(x)$ *for every* $x \in S$*. In other words,* x^* *is a global minimum.*

Proof: (by contradiction) Suppose x^* is a local but not global minimum.

Since x^* is a local optima, there exists $\epsilon > 0$ so that $f(x^*) \leq f(x)$ for all $x \in S$, $||x - x^*||_2 \leq \epsilon$.

Since x^* is not a global minimum, we can find $x_0 \in S$ where $f(x_0) < f(x^*)$.

Since S is convex, $\alpha x^* + (1-\alpha)x_0 \in S$ for every $\alpha \in [0,1]$.

Note that $f((1-\alpha)x^* + \alpha x_0) \leq (1-\alpha)f(x^*) + \alpha f(x_0) < f(x^*)$.

Pick $\alpha' = \frac{\epsilon}{2||x^* - x_0||_2}$ and set $x' := (1-\alpha')x^* + \alpha' x_0$.

Convex Program: Local Optima are Global Optima

Theorem (Equivalence of Local and Global Optima)

Let $\min_{x \in S} f(x)$ *be a convex program. If* x^* *is a local minimum, then* $f(x^*) \leq f(x)$ *for every* $x \in S$*. In other words,* x^* *is a global minimum.*

Proof: (by contradiction) Suppose x^* is a local but not global minimum.

Since x^* is a local optima, there exists $\epsilon > 0$ so that $f(x^*) \leq f(x)$ for all $x \in S$, $||x - x^*||_2 \leq \epsilon$.

Since x^* is not a global minimum, we can find $x_0 \in S$ where $f(x_0) < f(x^*)$.

Since S is convex, $\alpha x^* + (1 - \alpha)x_0 \in S$ for every $\alpha \in [0, 1]$.

Note that $f((1 - \alpha)x^* + \alpha x_0) \leq (1 - \alpha)f(x^*) + \alpha f(x_0) < f(x^*)$.

Pick $\alpha' = \frac{\epsilon}{2||x^* - x_0||_2}$ and set $x' := (1 - \alpha')x^* + \alpha' x_0$.

We have $f(x') < f(x^*)$ and $||x^* - x'||_2 \leq \epsilon$.

Convex Program: Local Optima are Global Optima

Theorem (Equivalence of Local and Global Optima)

Let $\min_{x \in S} f(x)$ be a convex program. If x^* is a local minimum, then $f(x^*) \leq f(x)$ for every $x \in S$. In other words, x^* is a global minimum.

Proof: (by contradiction) Suppose x^* is a local but not global minimum.

Since x^* is a local optima, there exists $\epsilon > 0$ so that $f(x^*) \leq f(x)$ for all $x \in S$, $||x - x^*||_2 \leq \epsilon$.

Since x^* is not a global minimum, we can find $x_0 \in S$ where $f(x_0) < f(x^*)$.

Since S is convex, $\alpha x^* + (1 - \alpha)x_0 \in S$ for every $\alpha \in [0, 1]$.

Note that $f((1 - \alpha)x^* + \alpha x_0) \leq (1 - \alpha)f(x^*) + \alpha f(x_0) < f(x^*)$.

Pick $\alpha' = \frac{\epsilon}{2||x^* - x_0||_2}$ and set $x' := (1 - \alpha')x^* + \alpha' x_0$.

We have $f(x') < f(x^*)$ and $||x^* - x'||_2 \leq \epsilon$.

This contradicts the fact that x^* is a local minimum. $\square$

Linear Matrix Inequalities (LMI)

Goal: Introduce notation to efficiently express convex constraints.

Linear Matrix Inequalities (LMI)

Goal: Introduce notation to efficiently express convex constraints.

Definition (Vector Inequality)

For $x, y \in \mathbb{R}^d$, we use $x \preceq y$ to denote that x is **element-wise less than** y. Concretely, $x \preceq y$ if for every $1 \leq i \leq d$, $x_i \leq y_i$.

Linear Matrix Inequalities (LMI)

Goal: Introduce notation to efficiently express convex constraints.

Definition (Vector Inequality)

For $x, y \in \mathbb{R}^d$, we use $x \preceq y$ to denote that x is **element-wise less than** y. Concretely, $x \preceq y$ if for every $1 \leq i \leq d$, $x_i \leq y_i$.

Example: $x \succeq 0$ means all entries of x are non-negative.

Linear Matrix Inequalities (LMI)

Goal: Introduce notation to efficiently express convex constraints.

Definition (Vector Inequality)

For $x, y \in \mathbb{R}^d$, we use $x \preceq y$ to denote that x is **element-wise less than** y. Concretely, $x \preceq y$ if for every $1 \leq i \leq d$, $x_i \leq y_i$.

Example: $x \succeq 0$ means all entries of x are non-negative.

We can also use inequalities to define sets: $\{x : x \preceq y\}$.

Linear Matrix Inequalities (LMI)

Goal: Introduce notation to efficiently express convex constraints.

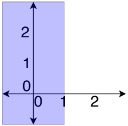
Definition (Vector Inequality)

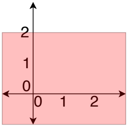
For $x, y \in \mathbb{R}^d$, we use $x \preceq y$ to denote that x is **element-wise less than** y. Concretely, $x \preceq y$ if for every $1 \leq i \leq d$, $x_i \leq y_i$.

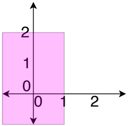
Example: $x \succeq 0$ means all entries of x are non-negative.

We can also use inequalities to define sets: $\{x : x \preceq y\}$.

Example: $\left\{ x : x \preceq \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\}$







$x_1 \leq 1$ $\qquad$ $x_2 \leq 2$ $\qquad$ $x \preceq (1, 2)^\top$

Matrix Inequalities

Definition (Positive Semidefinite Matrices)

We say a matrix $A \in \mathbb{R}^{d \times d}$ is positive semidefinite if $x^\top A x \geq 0$ for every $x \in \mathbb{R}^d$. The relation $A \succeq 0$ is often used to denote positive semidefiniteness of A.

Matrix Inequalities

Definition (Positive Semidefinite Matrices)

We say a matrix $A \in \mathbb{R}^{d\times d}$ is positive semidefinite if $x^\top Ax \geq 0$ for every $x \in \mathbb{R}^d$. The relation $A \succeq 0$ is often used to denote positive semidefiniteness of A.

Definition (Matrix Inequalities)

We say $A \preceq B$ if $0 \preceq B - A$, i.e. $B - A$ is positive semidefinite.

Matrix Inequalities

Definition (Positive Semidefinite Matrices)

We say a matrix $A \in \mathbb{R}^{d \times d}$ is positive semidefinite if $x^\top A x \geq 0$ for every $x \in \mathbb{R}^d$. The relation $A \succeq 0$ is often used to denote positive semidefiniteness of A.

Definition (Matrix Inequalities)

We say $A \preceq B$ if $0 \preceq B - A$, i.e. $B - A$ is positive semidefinite.

The set $\{A : A \succeq 0\}$ is a convex set (in fact, it is a cone). Optimizations of convex functions over this set are **Semidefinite Programs** (SDP).

Matrix Inequalities

Definition (Positive Semidefinite Matrices)

We say a matrix $A \in \mathbb{R}^{d \times d}$ is positive semidefinite if $x^\top Ax \geq 0$ for every $x \in \mathbb{R}^d$. The relation $A \succeq 0$ is often used to denote positive semidefiniteness of A.

Definition (Matrix Inequalities)

We say $A \preceq B$ if $0 \preceq B - A$, i.e. $B - A$ is positive semidefinite.

The set $\{A : A \succeq 0\}$ is a convex set (in fact, it is a cone). Optimizations of convex functions over this set are **Semidefinite Programs** (SDP).

Applications of SDPs: Sum of Squares Programming, Lyapunov Stability analysis, approximation algorithms for combinatorial optimization.

Optimization Models and Tools

Optimization Software

CPLEX

- Linear Programming (LP).
- Quadratic Programming (QP).
- Mixed-Integer Linear Programming (MILP).
- Mixed-Integer Quadratic Programming (MIQP).

Optimization Software

CPLEX

- Linear Programming (LP).
- Quadratic Programming (QP).
- Mixed-Integer Linear Programming (MILP).
- Mixed-Integer Quadratic Programming (MIQP).

`CVX`

- Linear Programming.
- Quadratic Programming.
- Semidefinite Programming.
- Convex Programming.
- Mixed-Integer Linear Programming (MILP).
- Mixed-Integer Quadratic Programming (MIQP).

Linear Programming

Goal: Minimize a linear function subject to linear equality and inequality constraints.

Linear Programming

Goal: Minimize a linear function subject to linear equality and inequality constraints. Mathematically,

$$
\begin{aligned}
\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^\top x \\
\text{subject to} \quad & Ax \preceq b \\
& A_{eq} x = b_{eq}.
\end{aligned}
$$

Goal: Minimize a linear function subject to linear equality and inequality constraints. Mathematically,

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}}\ & c^\top x \\ \text{subject to}\ & Ax \preceq b \\ & A_{eq} x = b_{eq}. \end{aligned}$$

A linear programming instance is specified by
$c \in \mathbb{R}^n, b \in \mathbb{R}^p, A \in \mathbb{R}^{p \times n}, b_{eq} \in \mathbb{R}^q, A_{eq} \in \mathbb{R}^{q \times n}$.

Linear Programming

Goal: Minimize a linear function subject to linear equality and inequality constraints. Mathematically,

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \ & c^\top x \\ \text{subject to} \ & Ax \preceq b \\ & A_{eq}x = b_{eq}. \end{aligned}$$

A linear programming instance is specified by
$c \in \mathbb{R}^n, b \in \mathbb{R}^p, A \in \mathbb{R}^{p \times n}, b_{eq} \in \mathbb{R}^q, A_{eq} \in \mathbb{R}^{q \times n}$.

Software:

CPLEX: `x = cplexlp(c, A, b, Aeq, beq)`.

MATLAB: `x = linprog(c, A,b, Aeq, beq)`.

LP Example - Marketplace Efficiency

Consider a market with n buyers $\{b_1, b_2, ..., b_n\}$ and m sellers $\{s_1, s_2, ..., s_m\}$.

LP Example - Marketplace Efficiency

Consider a market with n buyers $\{b_1, b_2, ..., b_n\}$ and m sellers $\{s_1, s_2, ..., s_m\}$.

Each buyer wants to buy at most 1 item.

LP Example - Marketplace Efficiency

Consider a market with n buyers $\{b_1, b_2, ..., b_n\}$ and m sellers $\{s_1, s_2, ..., s_m\}$.

Each buyer wants to buy at most 1 item.

Each seller has one item for sale.

LP Example - Marketplace Efficiency

Consider a market with n buyers $\{b_1, b_2, ..., b_n\}$ and m sellers $\{s_1, s_2, ..., s_m\}$.

- Each buyer wants to buy at most 1 item.
- Each seller has one item for sale.
- u_{ij} is the utility achieved when b_i buys from s_j.

LP Example - Marketplace Efficiency

Consider a market with n buyers $\{b_1, b_2, ..., b_n\}$ and m sellers $\{s_1, s_2, ..., s_m\}$.

- Each buyer wants to buy at most 1 item.
- Each seller has one item for sale.
- u_{ij} is the utility achieved when b_i buys from s_j.

Objective: Match buyers to sellers to maximize the total utility of the marketplace.

LP Example - Marketplace Efficiency

Graph Representation:

Construct a graph where the vertices are $\{b_1, \dots, b_n, s_1, \dots, s_m\}$.

LP Example - Marketplace Efficiency

Graph Representation:

Construct a graph where the vertices are $\{b_1, ..., b_n, s_1, ..., s_m\}$.
Include an edge between b_i and s_j of weight u_{ij} if $u_{ij} > 0$.

LP Example - Marketplace Efficiency

Graph Representation:

Construct a graph where the vertices are $\{b_1, ..., b_n, s_1, ..., s_m\}$.

Include an edge between b_i and s_j of weight u_{ij} if $u_{ij} > 0$.

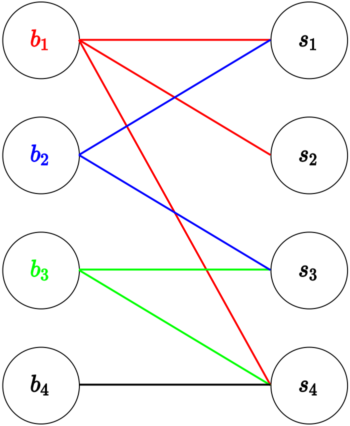
Finding the maximum utility matching becomes a maximum weight bipartite matching problem in this graph!

Graph Representation:

Construct a graph where the vertices are $\{b_1, ..., b_n, s_1, ..., s_m\}$.

Include an edge between b_i and s_j of weight u_{ij} if $u_{ij} > 0$.

Finding the maximum utility matching becomes a maximum weight bipartite matching problem in this graph!



LP Example - Marketplace Efficiency

Cast the maximum weight bipartite matching problem as a linear program:

LP Example - Marketplace Efficiency

Cast the maximum weight bipartite matching problem as a linear program:

Decision variable: $x \in \mathbb{R}^{mn}$, where x_{ij} determines whether or not b_i will buy from s_j.

LP Example - Marketplace Efficiency

Cast the maximum weight bipartite matching problem as a linear program:

Decision variable: $x \in \mathbb{R}^{mn}$, where x_{ij} determines whether or not b_i will buy from s_j.

$$
\begin{aligned}
\underset{x \in \mathbb{R}^{mn}}{\text{maximize}} \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} u_{ij} x_{ij} && (1) \\
\text{subject to} \quad & \sum_{j=1}^{m} x_{ij} \leq 1 \text{ for all } 1 \leq i \leq n && (2) \\
& \sum_{i=1}^{n} x_{ij} \leq 1 \text{ for all } 1 \leq j \leq m && (3) \\
& x \succeq 0. && (4)
\end{aligned}
$$

(2) ensures each buyer buys at most one item, (3) ensures each seller sells at most one item.

Even though fractional solutions are feasible for (1), we can always find an optimal solution which is integral $x^* \in \{0, 1\}^{mn}$!

LP Example - Marketplace Efficiency

Even though fractional solutions are feasible for (1), we can always find an optimal solution which is integral $x^* \in \{0, 1\}^{mn}$!

If $x_{ij}^* = 1$, have buyer b_i buy from seller s_j.

LP Example - Marketplace Efficiency

Even though fractional solutions are feasible for (1), we can always find an optimal solution which is integral $x^* \in \{0,1\}^{mn}$!

If $x^*_{ij} = 1$, have buyer b_i buy from seller s_j.

Remark: If we look at the KKT conditions of LP (1), the dual variables of the constraints can be used as prices with the following property:

> If the sellers s_j lists their item for a price λ_j,
>
> Buyer i chooses to buy the item $\arg\max_j u_{ij} - \lambda_j$,

then the resulting allocation will be x^*!

Linear Programming - Properties

Linear programs can be solved efficiently (millions of variables and constraints); They are among the easiest convex optimization problems to solve.

Linear Programming - Properties

Linear programs can be solved efficiently (millions of variables and constraints); They are among the easiest convex optimization problems to solve.

There are many applications: Pattern planning, minimum weight matching, multi-commodity maximum flow, production planning, etc.

Linear Programming - Properties

Linear programs can be solved efficiently (millions of variables and constraints); They are among the easiest convex optimization problems to solve.

There are many applications: Pattern planning, minimum weight matching, multi-commodity maximum flow, production planning, etc.

Definition (Extreme Point)

Given a convex set S, a point x is called extreme if it cannot be written as a convex combination of other points in S.

Linear Programming - Properties

Linear programs can be solved efficiently (millions of variables and constraints); They are among the easiest convex optimization problems to solve.

There are many applications: Pattern planning, minimum weight matching, multi-commodity maximum flow, production planning, etc.

Definition (Extreme Point)

Given a convex set S, a point x is called extreme if it cannot be written as a convex combination of other points in S.

As a consequence, all points in S can be written as convex combinations of the extreme points of S.

Linear Programming - Properties

For a linear program, the constraint set is comprised of linear equality and inequality constraints.

Linear Programming - Properties

For a linear program, the constraint set is comprised of linear equality and inequality constraints.

This means the constraint set is a polyhedron.

Linear Programming - Properties

For a linear program, the constraint set is comprised of linear equality and inequality constraints.

This means the constraint set is a polyhedron.

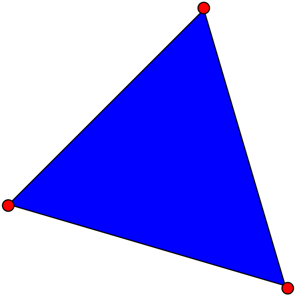
Extreme points of polyhedra are the corners.

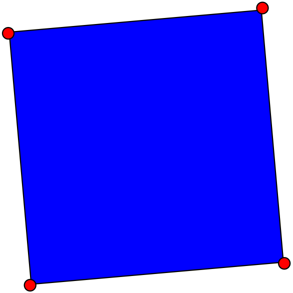
Linear Programming - Properties

For a linear program, the constraint set is comprised of linear equality and inequality constraints.

This means the constraint set is a polyhedron.

Extreme points of polyhedra are the corners.





Linear Programming - Properties

Theorem (Extreme Solutions of Linear Programs)

If a linear program $\min_{x \in P} c^\top x$ *has a finite optimal value (i.e. it has a non-empty solution set), then the solution set contains at least one extreme point of* P.

Linear Programming - Properties

Theorem (Extreme Solutions of Linear Programs)

If a linear program $\min_{x \in P} c^\top x$ has a finite optimal value (i.e. it has a non-empty solution set), then the solution set contains at least one extreme point of P.

Proof: Let $x^* \in P$ be an optimal solution.

Linear Programming - Properties

Theorem (Extreme Solutions of Linear Programs)

If a linear program $\min_{x \in P} c^\top x$ has a finite optimal value (i.e. it has a non-empty solution set), then the solution set contains at least one extreme point of P.

Proof: Let $x^* \in P$ be an optimal solution.

Let E_P be the set of extreme points of P.

Linear Programming - Properties

Theorem (Extreme Solutions of Linear Programs)

If a linear program $\min_{x \in P} c^\top x$ *has a finite optimal value (i.e. it has a non-empty solution set), then the solution set contains at least one extreme point of* P*.*

Proof: Let $x^* \in P$ be an optimal solution.

Let E_P be the set of extreme points of P.

Since $x^* \in P$, we can write it as a convex combination of points in E_P.

Linear Programming - Properties

Theorem (Extreme Solutions of Linear Programs)

If a linear program $\min_{x \in P} c^\top x$ *has a finite optimal value (i.e. it has a non-empty solution set), then the solution set contains at least one extreme point of* P*.*

Proof: Let $x^* \in P$ be an optimal solution.

Let E_P be the set of extreme points of P.

Since $x^* \in P$, we can write it as a convex combination of points in E_P.

Thus $x^* = \sum_{x \in E_P} \alpha_x x$ where $\sum_{x \in E_P} \alpha_x = 1$ and $\alpha_x \geq 0$.

Linear Programming - Properties

Theorem (Extreme Solutions of Linear Programs)

If a linear program $\min_{x \in P} c^\top x$ *has a finite optimal value (i.e. it has a non-empty solution set), then the solution set contains at least one extreme point of* P.

Proof: Let $x^* \in P$ be an optimal solution.

Let E_P be the set of extreme points of P.

Since $x^* \in P$, we can write it as a convex combination of points in E_P.

Thus $x^* = \sum_{x \in E_P} \alpha_x x$ where $\sum_{x \in E_P} \alpha_x = 1$ and $\alpha_x \geq 0$.

Thus $c^\top x^* = \sum_{x \in E_P} \alpha_x c^\top x \geq \min_{x \in E_P} c^\top x$, since the minimum is always at most the average.

Linear Programming - Properties

Theorem (Extreme Solutions of Linear Programs)

If a linear program $\min_{x \in P} c^\top x$ has a finite optimal value (i.e. it has a non-empty solution set), then the solution set contains at least one extreme point of P.

Proof: Let $x^* \in P$ be an optimal solution.

Let E_P be the set of extreme points of P.

Since $x^* \in P$, we can write it as a convex combination of points in E_P.

Thus $x^* = \sum_{x \in E_P} \alpha_x x$ where $\sum_{x \in E_P} \alpha_x = 1$ and $\alpha_x \geq 0$.

Thus $c^\top x^* = \sum_{x \in E_P} \alpha_x c^\top x \geq \min_{x \in E_P} c^\top x$, since the minimum is always at most the average.

So there is some $x' \in E_P$ with $c^\top x' \leq c^\top x^*$.

Linear Programming - Properties

Theorem (Extreme Solutions of Linear Programs)

If a linear program $\min_{x \in P} c^\top x$ *has a finite optimal value (i.e. it has a non-empty solution set), then the solution set contains at least one extreme point of* P*.*

Proof: Let $x^* \in P$ be an optimal solution.

Let E_P be the set of extreme points of P.

Since $x^* \in P$, we can write it as a convex combination of points in E_P.

Thus $x^* = \sum_{x \in E_P} \alpha_x x$ where $\sum_{x \in E_P} \alpha_x = 1$ and $\alpha_x \geq 0$.

Thus $c^\top x^* = \sum_{x \in E_P} \alpha_x c^\top x \geq \min_{x \in E_P} c^\top x$, since the minimum is always at most the average.

So there is some $x' \in E_P$ with $c^\top x' \leq c^\top x^*$.

Since x^* is a minimizer, x' must also be a minimizer.

Linear Programming - Properties

Theorem (Extreme Solutions of Linear Programs)

If a linear program $\min_{x \in P} c^\top x$ *has a finite optimal value (i.e. it has a non-empty solution set), then the solution set contains at least one extreme point of* P.

Note that this does NOT mean that every optimal point is extreme!

Linear Programming - Properties

Theorem (Extreme Solutions of Linear Programs)

If a linear program $\min_{x \in P} c^\top x$ *has a finite optimal value (i.e. it has a non-empty solution set), then the solution set contains at least one extreme point of* P.

Note that this does NOT mean that every optimal point is extreme!

However, this fact motivated the first implementation of an LP solver.

LP Algorithm: Simplex Method

Key Idea: Visit extreme points of the feasible region until you find the optimal solution.

LP Algorithm: Simplex Method

Key Idea: Visit extreme points of the feasible region until you find the optimal solution.

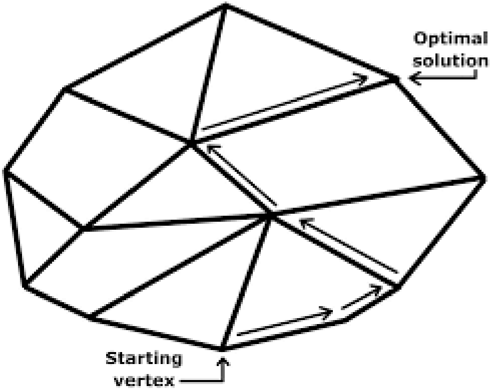


Figure taken from UC Davis

LP Algorithm: Simplex Method

Key Idea: Visit extreme points of the feasible region until you find the optimal solution.

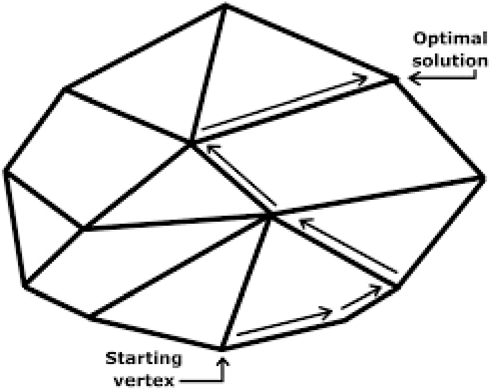


Figure taken from UC Davis

Remark: Worst case running time is exponential in the size of the input, but the set of "bad instances" is very small. In practice, the algorithm works well.

Quadratic Programming

Goal: Minimize a quadratic function subject to linear constraints.

Quadratic Programming

Goal: Minimize a quadratic function subject to linear constraints. Mathematically,

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & \frac{1}{2} x^\top H x + f^\top x \\ \text{subject to} \quad & Ax \preceq b \\ & A_{eq} x = b_{eq} \end{aligned}$$

where $H \succeq 0$.

Quadratic Programming

Goal: Minimize a quadratic function subject to linear constraints. Mathematically,

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \; & \frac{1}{2} x^\top H x + f^\top x \\ \text{subject to } & Ax \preceq b \\ & A_{eq} x = b_{eq} \end{aligned}$$

where $H \succeq 0$.

A quadratic programming instance is specified by
$f \in \mathbb{R}^n, H \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^p, A \in \mathbb{R}^{p \times n}, b_{eq} \in \mathbb{R}^q, A_{eq} \in \mathbb{R}^{q \times n}$.

Quadratic Programming

Goal: Minimize a quadratic function subject to linear constraints. Mathematically,

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \ & \frac{1}{2} x^\top H x + f^\top x \\ \text{subject to} \ & Ax \preceq b \\ & A_{eq} x = b_{eq} \end{aligned}$$

where $H \succeq 0$.

A quadratic programming instance is specified by $f \in \mathbb{R}^n, H \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^p, A \in \mathbb{R}^{p \times n}, b_{eq} \in \mathbb{R}^q, A_{eq} \in \mathbb{R}^{q \times n}$.

Software:

CPLEX: `x = cplexqp(H, f, A, b, Aeq, beq)`.

MATLAB: `x = quadprog(H, f, A,b, Aeq, beq)`.

QP Example: Discrete LQR

The discrete Linear Quadratic Regulator (LQR) with control effort constraints u_{LB}, u_{UB} can be formulated as a QP.

QP Example: Discrete LQR

The discrete Linear Quadratic Regulator (LQR) with control effort constraints u_{LB}, u_{UB} can be formulated as a QP.

$$
\begin{aligned}
\underset{u\in\mathbb{R}^T}{\text{minimize}} \; & \frac{1}{2}x_T^\top Q_T x_T + \frac{1}{2}\sum_{t=0}^{T-1} x_t^\top Q x_t + u_t^\top R u_t && (5)\\
\text{subject to } & x_{t+1} = Ax_t + Bu_t \text{ for all } 0 \le t \le T-1 && (6)\\
& x_0 = \text{initial condition} && (7)\\
& u_{LB} \preceq u_t \preceq u_{UB} \text{ for all } 0 \le t \le T-1. && (8)
\end{aligned}
$$

Optimization Algorithms

Algorithms for Convex Optimization

Simplex Method (Recap)

- Can solve linear programs.
- Was the first proposed algorithm to solve linear programs.
- Has a worst-case running time that is exponential in the input size.
- However, these examples are "pathological" and are rare.

Algorithms for Convex Optimization

Simplex Method (Recap)

- Can solve linear programs.
- Was the first proposed algorithm to solve linear programs.
- Has a worst-case running time that is exponential in the input size.
- However, these examples are "pathological" and are rare.

Ellipsoid Method

- Can solve convex problems.
- Was the first polynomial time algorithm for LP.
- Based on cutting plane techniques.
- Can handle exponentially many constraints, provided that feasibility can be queried in polynomial time.

Algorithms for Convex Optimization

Simplex Method (Recap)

- Can solve linear programs.
- Was the first proposed algorithm to solve linear programs.
- Has a worst-case running time that is exponential in the input size.
- However, these examples are "pathological" and are rare.

Ellipsoid Method

- Can solve convex problems.
- Was the first polynomial time algorithm for LP.
- Based on cutting plane techniques.
- Can handle exponentially many constraints, provided that feasibility can be queried in polynomial time.

Interior Point Methods

- Can solve convex problems.
- Based on sequential convex programming and warm-starts.
- Currently the best algorithm for general linear programming.

Ellipsoid Method - Cutting Plane Methods

Suppose we want to solve

$$\begin{aligned} &\underset{x \in \mathbb{R}^d}{\text{minimize}} \; f(x) \\ &\text{subject to } g_i(x) \leq 0 \text{ for } 1 \leq i \leq m. \end{aligned}$$

where $f, \{g_i\}_{i=1}^m$ are convex.

Ellipsoid Method - Cutting Plane Methods

Suppose we want to solve

$$\begin{aligned} &\underset{x \in \mathbb{R}^d}{\text{minimize}}\ f(x) \\ &\text{subject to } g_i(x) \leq 0 \text{ for } 1 \leq i \leq m. \end{aligned}$$

where $f, \{g_i\}_{i=1}^m$ are convex.

Key Idea: Every differentiable convex function $f : \mathbb{R}^d \to \mathbb{R}$ satisfies

$$f(y) - f(x) \geq \nabla f(x)^\top (y - x). \tag{9}$$

Ellipsoid Method - Cutting Plane Methods

Suppose we want to solve

$$\begin{aligned} &\underset{x \in \mathbb{R}^d}{\text{minimize}} \ f(x) \\ &\text{subject to } g_i(x) \leq 0 \text{ for } 1 \leq i \leq m. \end{aligned}$$

where $f, \{g_i\}_{i=1}^m$ are convex.

Key Idea: Every differentiable convex function $f : \mathbb{R}^d \to \mathbb{R}$ satisfies

$$f(y) - f(x) \geq \nabla f(x)^\top (y - x). \tag{9}$$

Therefore, all points in the halfplane $\left\{y : \nabla f(x)^\top (y - x) > 0\right\}$ have a larger objective value than x.

Ellipsoid Method - Cutting Plane Methods

Suppose we want to solve

$$\begin{aligned} &\underset{x \in \mathbb{R}^d}{\text{minimize}} \; f(x) \\ &\text{subject to } g_i(x) \le 0 \text{ for } 1 \le i \le m. \end{aligned}$$

where $f, \{g_i\}_{i=1}^m$ are convex.

Key Idea: Every differentiable convex function $f : \mathbb{R}^d \to \mathbb{R}$ satisfies

$$f(y) - f(x) \ge \nabla f(x)^\top (y - x). \tag{9}$$

Therefore, all points in the halfplane $\left\{y : \nabla f(x)^\top (y - x) > 0\right\}$ have a larger objective value than x.

Thus the global minimum cannot be in the set $\left\{y : \nabla f(x)^\top (y - x) > 0\right\}$. Therefore we can eliminate/"cut away" this set from our search space. Hence the name, "cutting plane method".

Ellipsoid Method - Cutting Plane Methods

Similarly, if $g_i(x) > 0$, then by convexity of g_i, we know

$$g_i(y) - g_i(x) \geq \nabla g_i(x)^\top (y - x). \tag{10}$$

Similarly, if $g_i(x) > 0$, then by convexity of g_i, we know

$$g_i(y) - g_i(x) \geq \nabla g_i(x)^\top (y - x). \tag{10}$$

In particular, all points in $\left\{y : \nabla g_i(x)^\top (y - x) > 0\right\}$ are infeasible! So we can prune away this set.

Ellipsoid Method - Bounding Ellipses

The Ellipsoid Method

Initialize $\mathcal{E}$ to be a large ball that contains the entire feasible set.

Ellipsoid Method - Bounding Ellipses

The Ellipsoid Method

- Initialize $\mathcal{E}$ to be a large ball that contains the entire feasible set.
- Query the center of $\mathcal{E}$, denote that point as x.

Ellipsoid Method - Bounding Ellipses

The Ellipsoid Method

Initialize $\mathcal{E}$ to be a large ball that contains the entire feasible set.

Query the center of $\mathcal{E}$, denote that point as x.

If x is feasible, set $\mathcal{A} \leftarrow \left\{ y : \nabla f(x)^\top (y - x) > 0 \right\}$.

Ellipsoid Method - Bounding Ellipses

The Ellipsoid Method

Initialize $\mathcal{E}$ to be a large ball that contains the entire feasible set.

Query the center of $\mathcal{E}$, denote that point as x.

If x is feasible, set $\mathcal{A} \leftarrow \{y : \nabla f(x)^\top (y - x) > 0\}$.

If x is infeasible, find a constraint g_i that is violated. Set $\mathcal{A} \leftarrow \{y : \nabla g_i(x)^\top (y - x) > 0\}$.

Ellipsoid Method - Bounding Ellipses

The Ellipsoid Method

- Initialize $\mathcal{E}$ to be a large ball that contains the entire feasible set.
- Query the center of $\mathcal{E}$, denote that point as x.
- If x is feasible, set $\mathcal{A} \leftarrow \{y : \nabla f(x)^\top (y - x) > 0\}$.
- If x is infeasible, find a constraint g_i that is violated. Set $\mathcal{A} \leftarrow \{y : \nabla g_i(x)^\top (y - x) > 0\}$.
- Update $\mathcal{E}$ to be the smallest ellipse that contains $\mathcal{E} \setminus \mathcal{A}$.

Ellipsoid Method - Bounding Ellipses

The Ellipsoid Method

Initialize $\mathcal{E}$ to be a large ball that contains the entire feasible set.

Query the center of $\mathcal{E}$, denote that point as x.

If x is feasible, set $\mathcal{A} \leftarrow \{y : \nabla f(x)^\top (y - x) > 0\}$.

If x is infeasible, find a constraint g_i that is violated. Set $\mathcal{A} \leftarrow \{y : \nabla g_i(x)^\top (y - x) > 0\}$.

Update $\mathcal{E}$ to be the smallest ellipse that contains $\mathcal{E} \setminus \mathcal{A}$.

Terminate once $\text{vol}(\mathcal{E}) < \epsilon$ and output the center of $\mathcal{E}$.

The Ellipsoid Method

Initialize $\mathcal{E}$ to be a large ball that contains the entire feasible set.

Query the center of $\mathcal{E}$, denote that point as x.

If x is feasible, set $\mathcal{A} \leftarrow \{y : \nabla f(x)^\top (y - x) > 0\}$.

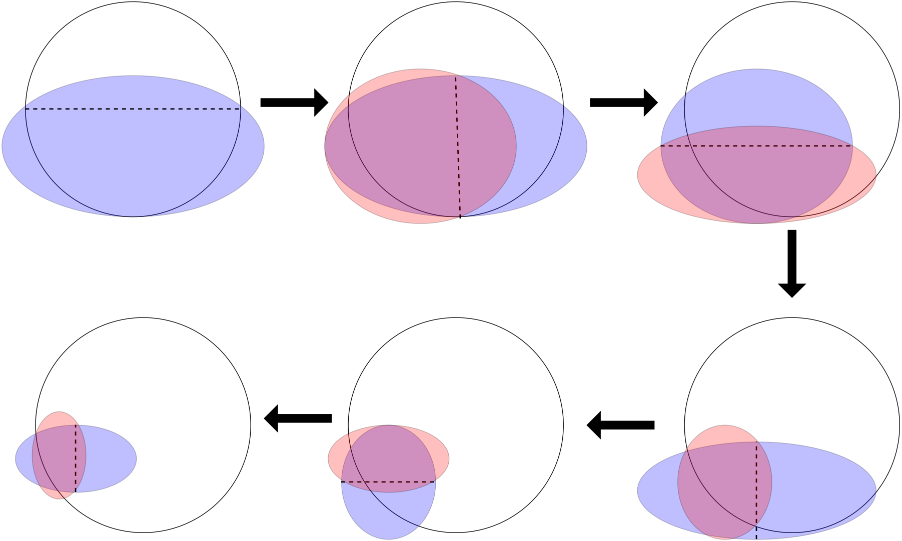
If x is infeasible, find a constraint g_i that is violated. Set $\mathcal{A} \leftarrow \{y : \nabla g_i(x)^\top (y - x) > 0\}$.

Update $\mathcal{E}$ to be the smallest ellipse that contains $\mathcal{E} \setminus \mathcal{A}$.

Terminate once $\text{vol}(\mathcal{E}) < \epsilon$ and output the center of $\mathcal{E}$.

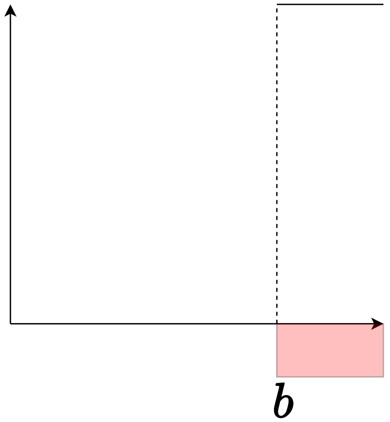
One can show that $\text{vol}(\mathcal{E})$ decreases fast enough in each iteration so that the algorithm will terminate in polynomial time with a high quality solution.

Ellipsoid Method - Visualization



Key Idea: Convert a constrained problem into an unconstrained problem using penalty functions.

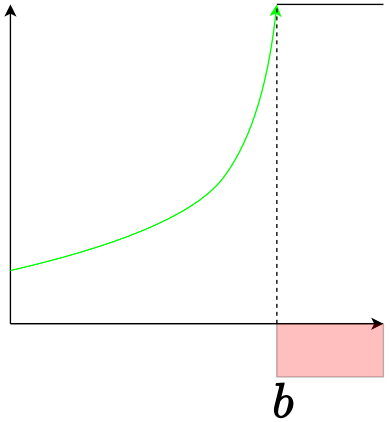
Example: Consider the constraint $x \leq b$.



Interior Point Methods

Key Idea: Convert a constrained problem into an unconstrained problem using penalty functions.

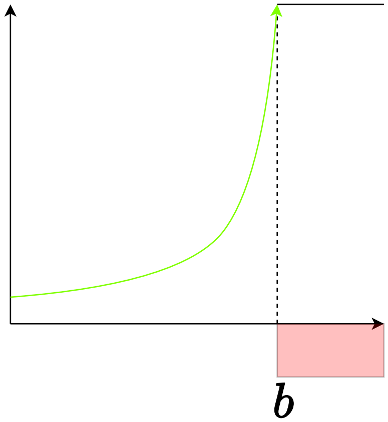
Example: Consider the constraint $x \leq b$.



Interior Point Methods

Key Idea: Convert a constrained problem into an unconstrained problem using penalty functions.

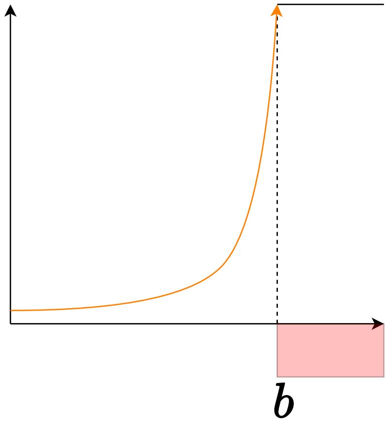
Example: Consider the constraint $x \leq b$.



Interior Point Methods

Key Idea: Convert a constrained problem into an unconstrained problem using penalty functions.

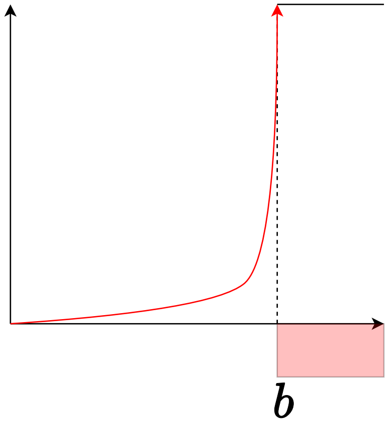
Example: Consider the constraint $x \leq b$.



Interior Point Methods

Key Idea: Convert a constrained problem into an unconstrained problem using penalty functions.

Example: Consider the constraint $x \leq b$.



Interior Point Methods - Barrier functions

Observation 1: Steep barriers are better approximations to boundary constraints, but they have poor conditioning that can lead to numerical instability.

Interior Point Methods - Barrier functions

Observation 1: Steep barriers are better approximations to boundary constraints, but they have poor conditioning that can lead to numerical instability.

Observation 2: Shallow/smooth barriers have good conditioning that leads to fast convergence, but they are bad at approximating the true constraint boundary.

Interior Point Methods - Barrier functions

Observation 1: Steep barriers are better approximations to boundary constraints, but they have poor conditioning that can lead to numerical instability.

Observation 2: Shallow/smooth barriers have good conditioning that leads to fast convergence, but they are bad at approximating the true constraint boundary.

Key Idea:

Provide a warm start to the problem with a steep barrier to overcome poor conditioning.

Interior Point Methods - Barrier functions

Observation 1: Steep barriers are better approximations to boundary constraints, but they have poor conditioning that can lead to numerical instability.

Observation 2: Shallow/smooth barriers have good conditioning that leads to fast convergence, but they are bad at approximating the true constraint boundary.

Key Idea:

Provide a warm start to the problem with a steep barrier to overcome poor conditioning.

Obtain a warm start point by solving the problem with a shallower barrier.

Interior Point Methods - Barrier functions

Observation 1: Steep barriers are better approximations to boundary constraints, but they have poor conditioning that can lead to numerical instability.

Observation 2: Shallow/smooth barriers have good conditioning that leads to fast convergence, but they are bad at approximating the true constraint boundary.

Key Idea:

- Provide a warm start to the problem with a steep barrier to overcome poor conditioning.
- Obtain a warm start point by solving the problem with a shallower barrier.
- Continue using each solution to warm start a problem with a steeper barrier until convergence.