LMI Methods in Optimal and Robust Control

Matthew M. Peet
Arizona State University

Lecture 02: Optimization (Convex and Otherwise)

Mathematical Optimization and Curly's Law

Curly: Do you know what the secret of life is?
Curly: One thing (metric). Just one thing. You stick to that (metric) and the rest don't mean ****.

$$\min_{x \in \mathbb{F}} \quad f(x): \qquad \text{subject to}$$
$$g_i(x) \le 0 \qquad i = 1, \cdots K_1$$
$$h_i(x) = 0 \qquad i = 1, \cdots K_2$$

Variables: $x \in \mathbb{F}$

- The things you must choose.
- $\mathbb{F}$ represents the set of possible choices for the variables.
- Can be vectors, matrices, functions, systems, locations, colors...
  - However, computers prefer vectors or matrices.

Objective: $f(x)$

- A function which assigns a *scalar* value to any choice of variables.
  - e.g. $[x_1, x_2] \mapsto x_1 - x_2$; red $\mapsto 4$; et c.

Constraints: $g(x) \le 0$; $h(x) = 0$

- Defines what is a minimally acceptable choice of variables (Feasible).

Formulating Optimization Problems

What do we need to know?

Topics to Cover:

Formulating Constraints

- Tricks of the Trade for expressing constraints.
- Converting everything to equality and inequality constraints.

Equivalence:

- How to Recognize if Two Optimization Problems are Equivalent.
- May be true despite different variables, constraints and objectives

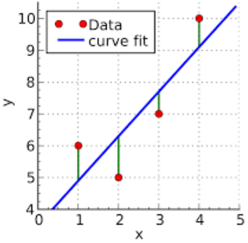
Knowing which Problems are Solvable

- The Convex Ones.
- Some others, if the problem is relatively small.

Problem: Given a bunch of data in the form

- Inputs: a_i
- Outputs: b_i

Find the function $f(a) = b$ which best fits the data.



For **Least Squares**: Assume $f(a) = z^T a + z_0$ where $z \in \mathbb{R}^n, z_0 \in \mathbb{R}$ are the variables with objective

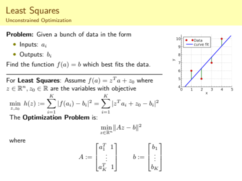
$$\min_{z,z_0} \; h(z) := \sum_{i=1}^{K} |f(a_i) - b_i|^2 = \sum_{i=1}^{K} |z^T a_i + z_0 - b_i|^2$$

The **Optimization Problem** is:

$$\min_{z \in \mathbb{R}^n} \|Az - b\|^2$$

where

$$A := \begin{bmatrix} a_1^T & 1 \\ & \vdots \\ a_K^T & 1 \end{bmatrix} \qquad b := \begin{bmatrix} b_1 \\ \vdots \\ b_K \end{bmatrix}$$



Least Squares
Unconstrained Optimization

Problem: Given a bunch of data in the form
- Inputs: a_i
- Outputs: b_i

Find the function $f(a) = b$ which best fits the data.

For **Least Squares**: Assume $f(a) = z^T a + z_0$ where $z \in \mathbb{R}^n, z_0 \in \mathbb{R}$ are the variables with objective

$$\min_{z,z_0} h(z) := \sum_{i=1}^{K} |f(a_i) - b_i|^2 = \sum_{i=1}^{K} |z^T a_i + z_0 - b_i|^2$$

The **Optimization Problem** is:

$$\min_{z \in \mathbb{R}^n} \|Az - b\|^2$$

where

$$A := \begin{bmatrix} a_1^T & 1 \\ & \vdots \\ a_K^T & 1 \end{bmatrix} \qquad b := \begin{bmatrix} b_1 \\ \vdots \\ b_K \end{bmatrix}$$

Boring/Conservative/Grumpy (Monarchist).

One of the greatest mathematicians

- Professor of Astronomy in Göttingen
- Motto: “pauca sed matura” (few but ripe)

Discovered

- Gaussian Distributions
- Gauss’ Law (collaboration with Weber)
- Non-Euclidean Geometry (maybe)
- Least Squares (maybe)

Legendre published the first solution to the Least Squares problem in 1805

- In typical fashion, Carl Friedrich Gauss claimed to have solved the problem in 1795 and published a more rigorous solution in 1809.
- This more rigorous solution first introduced the normal probability distribution (or Gaussian distribution)

Discovery and Rediscovery of Ceres

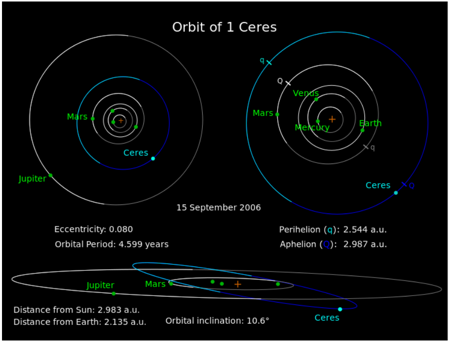
The pseudo-planet Ceres was discovered by G. Piazzi

- Observed 12 times between Jan. 1 and Feb. 11, 1801
- Planet was then lost.

Complication:

- Observation was only declination and right-ascension.
- Observations were only spread over 1% of the orbit.
  - No ranging info.

C. F. Gauss applied Least Squares and correctly predicted the location.



- Planet was re-found on Dec 31, 1801 in the correct location.

Solution to the Least Squares Problem

The **Least Squares Problem** is:

$$\min_{z \in \mathbb{R}^n} \|Az - b\|^2$$

where

$$A := \begin{bmatrix} a_1^T & 1 \\ \vdots & \\ a_K^T & 1 \end{bmatrix} \qquad b := \begin{bmatrix} b_1 \\ \vdots \\ b_K \end{bmatrix}$$

Least squares problems are easy-ish to solve.

$$z^* = (A^T A)^{-1} A^T b$$

Note that A is assumed to be skinny.

- More rows than columns.
- More data points than inputs (dimension of a_i is small).

The term $(A^T A)^{-1} A^T$ is referred to variously as

- The Moore-Penrose Inverse
- The pseudoinverse

Integer Programming Example

MAX-CUT

Optimization of a *graph*.

- Graphs have *Nodes* and *Edges*.

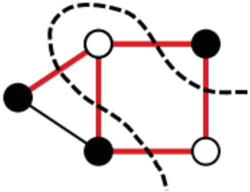


Figure: Division of a set of nodes to maximize the weighted cost of separation

Goal: Assign each node i an index $x_i = -1$ or $x_i = 1$ to maximize overall cost.

- The cost if x_i and x_j do not share the same index is w_{ij}.
- The cost if they share an index is 0
- The weights w_{ij} are given.

Integer Programming Example

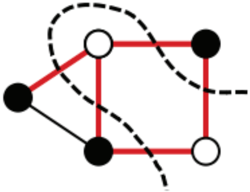
MAX-CUT

Goal: Assign each node i an index $x_i = -1$ or $x_j = 1$ to maximize overall cost.

Variables: $x \in \{-1, 1\}^n$

- Referred to as **Integer Variables** or **Binary Variables**.
- Binary constraints can be incorporated explicitly:

$$x_i^2 = 1$$



Integer/Binary variables may be declared directly in YALMIP:

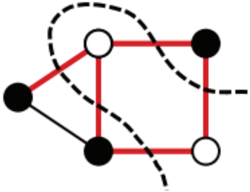
```
> x = intvar(n);
> y = binvar(n);
```

Integer Programming Example

MAX-CUT

Objective: We use the trick:

- $(1 - x_i x_j) = 0$ if x_i and x_j have the same sign (Together).
- $(1 - x_i x_j) = 2$ if x_i and x_j have the opposite sign (Apart).



Then the objective function is

$$\min \frac{1}{2} \sum_{i,j} w_{ij}(1 - x_i x_j)$$

The optimization problem is the integer program:

$$\max_{x_i^2=1} \frac{1}{2} \sum_{i,j} w_{ij}(1 - x_i x_j)$$

MAX-CUT

The optimization problem is the integer program:

$$\max_{x_i^2=1} \frac{1}{2}\sum_{i,j} w_{ij}(1 - x_i x_j)$$

Consider the MAX-CUT problem with 5 nodes

$$w_{12} = w_{23} = w_{45} = w_{15} = w_{34} = .5 \quad \text{and} \quad w_{14} = w_{24} = w_{25} = 0$$

where $w_{ij} = w_{ji}$.

An Optimal Cut IS:

- $x_1 = x_3 = x_4 = 1$
- $x_2 = x_5 = -1$

This cut has objective value

$$f(x) = 2.5 - .5x_1x_2 - .5x_2x_3 - .5x_3x_4 - .5x_4x_5 - .5x_1x_5 = 4$$

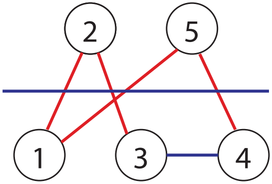


Figure: An Optimal Cut

Objective Function: Lets minimize a quadratic cost

$$x(N)^T S x(N) + \sum_{k=1}^{N-1} x(k)^T Q x(k) + u(k)^T R u(k)$$

Variables: The sequence of states $x(k)$, and inputs, $u(k)$.
Constraint: The dynamics define how $u \mapsto x$.

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \qquad k = 0, \cdots, N \\ x(0) &= 1 \end{aligned}$$

Optimization Formulation of DP:

$$\min_{x,u} \quad x(N)^T S x(N) + \sum_{k=1}^{N-1} \left(x(k)^T Q x(k) + u(k)^T R u(k) \right)$$

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \qquad k = 0, \cdots, N \\ x(0) &= 1 \end{aligned}$$

Optimization with Dynamics
Open-Loop Case (Dynamic Programming)

Objective Function: Lets minimize a quadratic cost

$$x(N)^T Sx(N) + \sum_{k=1}^{N-1} x(k)^T Qx(k) + u(k)^T Ru(k)$$

Variables: The sequence of states $x(k)$, and inputs, $u(k)$.
Constraint: The dynamics define how $u \mapsto x$.

$$x(k+1) = Ax(k) + Bu(k), \qquad k = 0, \cdots, N$$
$$x(0) = 1$$

Optimization Formulation of DP:

$$\min_{x,u} \quad x(N)^T Sx(N) + \sum_{k=1}^{N-1} \left(x(k)^T Qx(k) + u(k)^T Ru(k)\right)$$
$$x(k+1) = Ax(k) + Bu(k), \qquad k = 0, \cdots, N$$
$$x(0) = 1$$

Dynamic Programming has been around since the 1950's and can be solved recursively using Bellman's equation

- Actually, a nested sequence of optimization problems.
- Solution relies on the "Principle of Optimality"
- The principle of Optimality says that if we start anywhere along the optimal trajectory, that solution will still be optimal if we re-started the optimization problem from that point.
- Implies that the optimal input (for separable objectives) is always a function of the current state.
- The principle of optimality is also what underlies Djikstra's algorithm
- Djikstra's algorithm is what enables internet packet routing and the route-finding in Google (Apple) maps.

Objective Function: Lets minimize a quadratic Cost

$$x(N)^T S x(N) + \sum_{k=1}^{N-1} x(k)^T Q x(k) + u(k)^T R u(k)$$

Variables: We want a fixed *policy* (gain matrix, K) which determines $u(k)$ based on $x(k)$ as $u(k) = Kx(k)$.
Constraint: The dynamics define how $u \mapsto x$.

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \qquad k = 0, \cdots, N \\ u(k) &= Kx(k), \qquad x(0) = 1 \end{aligned}$$

Optimization Formulation of LQR:

$$\begin{aligned} \min_{x,u} \quad & x(N)^T S x(N) + \sum_{k=1}^{N-1} \left(x(k)^T Q x(k) + u(k)^T R u(k) \right) \\ x(k+1) &= Ax(k) + Bu(k), \qquad k = 0, \cdots, N \\ u(k) &= Kx(k), \qquad x(0) = 1 \end{aligned}$$

Question: Are the Closed-Loop and Open-Loop Problems Equivalent?

Optimization with Dynamics

Closed-Loop Case (LQR)

Objective Function: Lets minimize a quadratic Cost

$$x(N)^T S x(N) + \sum_{k=1}^{N-1} x(k)^T Q x(k) + u(k)^T R u(k)$$

Variables: We want a fixed *policy* (gain matrix, K) which determines $u(k)$ based on $x(k)$ as $u(k) = Kx(k)$.
Constraint: The dynamics define how $u \mapsto x$.

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \qquad k = 0, \cdots, N \\ u(k) &= Kx(k), \qquad x(0) = 1 \end{aligned}$$

Optimization Formulation of LQR:

$$\min_{x,u} \quad x(N)^T S x(N) + \sum_{k=1}^{N-1} \left(x(k)^T Q x(k) + u(k)^T R u(k) \right)$$

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \qquad k = 0, \cdots, N \\ u(k) &= Kx(k), \qquad x(0) = 1 \end{aligned}$$

Question: Are the Closed-Loop and Open-Loop Problems Equivalent?

LQR stands for Least Quadratic Regulator

- Least Quadratic refers to the quadratic cost function
- Regulator refers to feedback

By Equivalent, can we assume the optimal input is a static function of the current state?

- Bellman's equation says the optimal input for separable objectives is always a function of the current state.
- In the quadratic case, the resulting function is static

Definition 1.

Two optimization problems are **Equivalent** if a solution (algorithm/black box) to one can be used to construct a solution to the other.

Example 1: Equivalent Objective Functions

Problem 1: $\min_x f(x)$ subject to $A^T x \geq b$

Problem 2: $\min_x 10f(x) - 12$ subject to $A^T x \geq b$

Problem 3: $\max_x \frac{1}{f(x)}$ subject to $A^T x \geq b$

In this case $x_1^* = x_2^* = x_3^*$. Proof:

- For any $x \neq x_1^*$ (both feasible), since x_1^* is optimal, we have $f(x) > f(x_1^*)$. Thus $10f(x) - 12 > 10f(x_1^*) - 12$ and $\frac{1}{f(x)} < \frac{1}{f(x_1^*)}$. i.e x is suboptimal for all.

Definition 1.
Two optimization problems are **Equivalent** if a solution (algorithm/black box) to one can be used to construct a solution to the other.

Example 1: Equivalent Objective Functions

Problem 1:	$\min_x f(x)$	subject to	$A^T x \geq b$
Problem 2:	$\min_x 10f(x) - 12$	subject to	$A^T x \geq b$
Problem 3:	$\max_x \frac{1}{f(x)}$	subject to	$A^T x \geq b$

In this case $x_1^* = x_2^* = x_3^*$. Proof:

- For any $x \neq x_1^*$ (both feasible), since x_1^* is optimal, we have $f(x) > f(x_1^*)$. Thus $10f(x) - 12 > 10f(x_1^*) - 12$ and $\frac{1}{f(x)} < \frac{1}{f(x_1^*)}$. i.e x is suboptimal for all.

Here x_i^* is the solution to problem i

Example 2: Equivalent Variables

Problem 1: $\min_x f(x)$ subject to $A^T x \geq b$

Problem 2: $\min_x f(Tx + c)$ subject to $(T^T A)^T x \geq b - A^T c$

Here $x_1^* = T x_2^* + c$ and $x_2^* = T^{-1}(x_1^* - c)$.

- Change of variables is invertible. (given $x \neq x_2^*$, you can show it is suboptimal)

Example 3: Variable Separability

Problem 1: $\min_{x,y} f(x) + g(y)$ subject to $A_1^T x \geq b_1, A_2^T y \geq b_2$

Problem 2: $\min_x f(x)$ subject to $A_1^T x \geq b_1$

Problem 3: $\min_y g(y)$ subject to $A_2^T y \geq b_2$

Here $x_1^* = x_2^*$ and $y_1^* = y_3^*$.

- Neither feasibility nor minimality are coupled (Objective fn. is *Separable*).

Formulating Optimization Problems

Equivalence in Variables

Example 2: Equivalent Variables

Problem 1: $\min_x f(x)$ subject to $A^T x \geq b$

Problem 2: $\min_x f(Tx + c)$ subject to $(T^T A)^T x \geq b - A^T c$

Here $x_1^* = T x_2^* + c$ and $x_2^* = T^{-1}(x_1^* - c)$.

- Change of variables is invertible. (given $x \neq x_2^*$, you can show it is suboptimal)

Example 3: Variable Separability

Problem 1: $\min_{x,y} f(x) + g(y)$ subject to $A_1^T x \geq b_1, A_2^T y \geq b_2$

Problem 2: $\min_x f(x)$ subject to $A_1^T x \geq b_1$

Problem 3: $\min_y g(y)$ subject to $A_2^T y \geq b_2$

Here $x_1^* = x_2^*$ and $y_1^* = y_3^*$.

- Neither feasibility nor minimality are coupled (Objective fn. is *Separable*).

- If you add redundant variables (T is fat), the problems may still be equivalent.
- Variable Separability is what allows us to solve Dynamic Programming.

Example 4: Constraint/Objective Equivalence

Problem 1:	$\min_{x} f(x)$	subject to	$g(x) \leq 0$
Problem 2:	$\min_{x,t} t$	subject to	$g(x) \leq 0, \ t \geq f(x)$

Here $x_1^* = x_2^*$ and $t_2^* = f(x_1^*)$.

Some other Equivalences:

- Redundant Constraints
  - $\{x \in \mathbb{R} : \ x > 1\}$ vs. $\{x \in \mathbb{R} : \ x > 1, x > 0\}$
- Polytopes (Vertices vs. Hyperplanes)
  - $\{x \in \mathbb{R}^n : \ x = \sum_i A_i \alpha_i, \sum_i \alpha_i = 1\}$ vs. $\{x \in \mathbb{R}^n : \ Cx > b\}$

Formulating Optimization Problems

Constraint Equivalence

Example 4: Constraint/Objective Equivalence

Problem 1: $\min_x f(x)$ subject to $g(x) \leq 0$

Problem 2: $\min_{x,t} t$ subject to $g(x) \leq 0,\ t \geq f(x)$

Here $x_1^* = x_2^*$ and $t_2^* = f(x_1^*)$.

Some other Equivalences:

- Redundant Constraints
  - $\{x \in \mathbb{R} : x > 1\}$ vs. $\{x \in \mathbb{R} : x > 1, x > 0\}$
- Polytopes (Vertices vs. Hyperplanes)
  - $\{x \in \mathbb{R}^n : x = \sum_i A_i\alpha_i, \sum_i \alpha_i = 1\}$ vs. $\{x \in \mathbb{R}^n : Cx > b\}$

The set constraint $x \in S$ is what matters, not the *representation* of that set

In *Classification* we have inputs (data) (x_i), each of which has a binary label $(y_i \in \{-1, +1\})$

- $y_i = +1$ means the output of x_i belongs to group 1
- $y_i = -1$ means the output of x_i belongs to group 2

We want to find a rule (a classifier) which takes the data x and predicts which group it is in.

- Our rule has the form of a function $f(x) = w^T x - b$. Then
  - x is in group 1 if $f(x) = w^T x - b > 0$.
  - x is in group 2 if $f(x) = w^T x - b < 0$.

Question: How to find the best w and b??

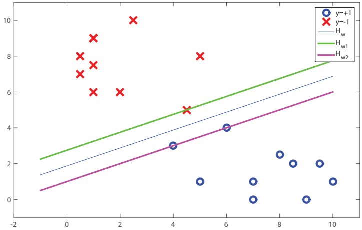
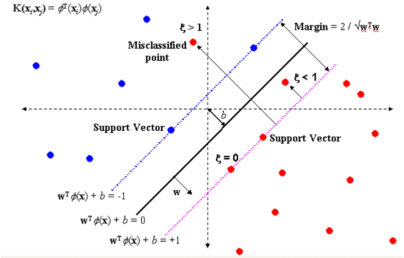


Figure: We want to find a rule which separates two sets of data.



Definition 2.

- A **Hyperplane** is the generalization of the concept of line/plane to multiple dimensions.
$$\{x \in \mathbb{R}^n \ : \ w^T x - b = 0\}$$
- **Half-Spaces** are the parts above and below a Hyperplane.
$$\{x \in \mathbb{R}^n \ : \ w^T x - b \geq 0\} \qquad \text{OR} \qquad \{x \in \mathbb{R}^n \ : \ w^T x - b \leq 0\}$$

We want to separate the data into disjoint half-spaces and maximize the distance between these half-spaces

Variables: $w \in \mathbb{R}^n$ and b define the hyperplane
Constraint: Each existing data point should be correctly labelled.

- $w^T x - b > 1$ when $y_i = +1$ and $w^T x - b < -1$ when $y_i = -1$ (Strict Separation)
- Alternatively: $y_i(w^T x_i - b) \geq 1$.

These two constraints are **Equivalent**.

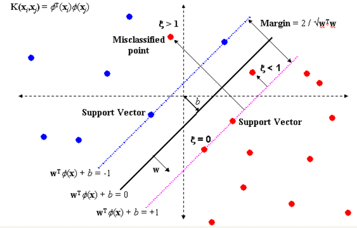


Figure: Maximizing the distance between two sets of Data

Objective: The distance between Hyperplanes $\{x : w^T x - b = 1\}$ and $\{x : w^T x - b = -1\}$ is

$$f(w,b) = 2\frac{1}{\sqrt{w^T w}}$$

Machine Learning algorithms solve

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} \frac{1}{2} w^T w, \qquad \text{subject to}$$

$$y_i(w^T x_i - b) \geq 1, \quad \forall i = 1, ..., K.$$

Soft Margin Problems

The hard margin problem can be relaxed to maximize the distance between hyperplanes PLUS the magnitude of classification errors

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} \frac{1}{2} \|w\|^2 + c \sum_{i=1}^{n} \max(0, 1 - (w^T x_i - b) y_i).$$

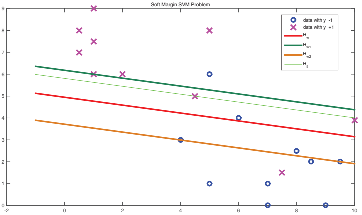


Figure: Data separation using soft-margin metric and distances to associated hyperplanes

Link: Repository of Interesting Machine Learning Data Sets

Geometric vs. Functional Constraints

These Problems are all equivalent:

The **Classical Representation**:

$$\min_{x\in\mathbb{R}^n} f(x): \qquad \text{subject to}$$
$$g_i(x) \le 0 \qquad i = 1, \cdots k$$

The **Geometric Representation** is:

$$\min_{x\in\mathbb{R}^n} f(x): \qquad \text{subject to} \qquad x \in S$$

where $S := \{x \in \mathbb{R}^n : g_i(x) \le 0,\ i = 1, \cdots, k\}$.

The **Pure Geometric Representation** (x is eliminated!):

$$\min_{\gamma} \gamma: \qquad \text{subject to}$$
$$S_\gamma \neq \emptyset \quad (S_\gamma \text{ has at least one element})$$

where $S_\gamma := \{x \in \mathbb{R}^n : \gamma - f(x) \ge 0,\ g_i(x) \le 0,\ i = 1, \cdots, k\}$.

Proposition: Optimization is only as hard as determining feasibility!

Geometric vs. Functional Constraints

These Problems are all equivalent:

The **Classical Representation**:

$$\min_{x \in \mathbb{R}^n} f(x): \qquad \text{subject to}$$
$$g_i(x) \le 0 \qquad i = 1, \cdots k$$

The **Geometric Representation** is:

$$\min_{x \in \mathbb{R}^n} f(x): \qquad \text{subject to} \qquad x \in S$$

where $S := \{x \in \mathbb{R}^n : g_i(x) \le 0,\ i = 1, \cdots, k\}$.

The **Pure Geometric Representation** (x is eliminated!):

$$\min_{\gamma} \gamma: \qquad \text{subject to}$$
$$S_\gamma \neq \emptyset \quad (S_\gamma \text{ has at least one element})$$

where $S_\gamma := \{x \in \mathbb{R}^n : \gamma - f(x) \ge 0,\ g_i(x) \le 0,\ i = 1, \cdots, k\}$.

Proposition: Optimization is only as hard as determining feasibility!

In the pure geometric interpretation, we are finding the smallest γ such that there exists a feasible point, x with $f(x) \le \gamma$

Solving by Bisection (Do you have an Oracle?)

Assume you can test feasibility of a set S_γ

Optimization Problem:

$$\gamma^* = \max_\gamma \ \gamma :$$

$$\text{subject to } S_\gamma \neq \emptyset$$

Bisection Algorithm (Convexity???):

1 Initialize infeasible $\gamma_u = b$

2 Initialize feasible $\gamma_l = a$

3 Set $\gamma = \frac{\gamma_u + \gamma_l}{2}$

5 If S_γ feasible, set $\gamma_l = \frac{\gamma_u + \gamma_l}{2}$

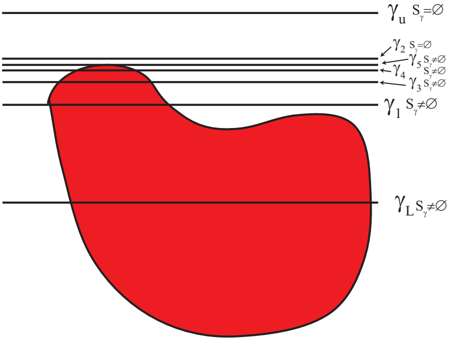
4 If S_γ infeasible, set $\gamma_u = \frac{\gamma_u + \gamma_l}{2}$

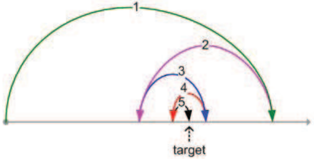
6 $k = k + 1$

7 Goto 3

Then $\gamma^* \in [\gamma_l, \gamma_u]$ and $|\gamma_u - \gamma_l| \leq \frac{b-a}{2^k}$.

Bisection with oracle also solves the Primary Problem. $(\min \gamma : \ S_\gamma = \emptyset)$





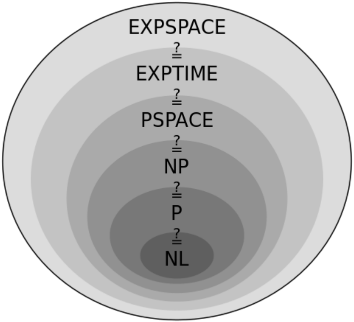
Computational Complexity

In Computer Science, we focus on **Complexity of the PROBLEM**

- NOT complexity of the algorithm.

On a Turing machine, the # of steps is a fn of problem size (number of variables)

- NL: A logarithmic # (SORT)
- P: A polynomial # (LP)
- NP: A polynomial # for verification (TSP)
- NP HARD: at least as hard as NP (TSP)
- NP COMPLETE: A set of Equivalent* NP problems (MAX-CUT, TSP)
- EXPTIME: Solvable in $2^{p(n)}$ steps. p polynomial. (Chess)
- EXPSPACE: Solvable with $2^{p(n)}$ memory.



*Equivalent means there is a polynomial-time reduction from one to the other.

How Hard is Optimization?

The **Classical Representation**:

$$\min_{x\in\mathbb{R}^n} f(x): \qquad \text{subject to}$$
$$g_i(x) \le 0 \qquad i = 1,\cdots k$$
$$h_i(x) = 0 \qquad i = 1,\cdots k$$

Answer: Easy (P) if f, g_i are all Convex and h_i are affine.

The **Geometric Representation**:

$$\min_{x\in\mathbb{R}^n} \; f(x): \qquad \text{subject to} \qquad x \in S$$

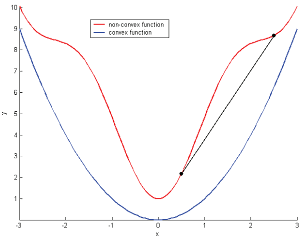
Answer: Easy (P) if f is Convex and S is a Convex Set.

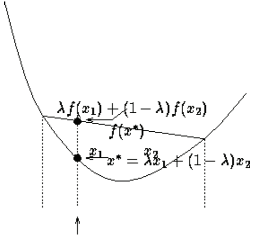
The **Pure Geometric Representation**:

$$\max_{\gamma, x\in\mathbb{R}^n} \; \gamma: \qquad \text{subject to}$$
$$(\gamma, x) \in S'$$

Answer: Easy (P) if S' is a Convex Set.

Convex Functions





Definition 3.

An OBJECTIVE FUNCTION or CONSTRAINT function is convex if $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$ for all $\lambda \in [0,1]$.

Useful Facts:

- e^{ax}, $\|x\|$ are convex. x^n ($n \geq 1$ or $n \leq 0$), $-\log x$ are convex on $x \geq 0$
- If f_1 is convex and f_2 is convex, then $f_3(x) := f_1(x) + f_2(x)$ is convex.
- A f is convex if the Hessian $\nabla^2 f(x)$ is positive semidefinite for all x.
- If f_1, f_2 are convex, then $f_3(x) := \max(f_1(x), f_2(x))$ is convex.
- If f_1, f_2 are convex, and f_1 is increasing, then $f_3(x) := f_1(f_2(x))$ is convex.

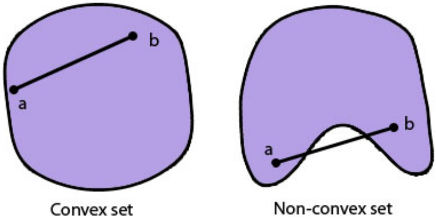
Convex Sets

Definition 4.

A FEASIBLE SET is **convex** if for any $x, y \in Q$,

$$\{\mu x + (1-\mu)y \,:\, \mu \in [0,1]\} \subset Q.$$

The line connecting any two points lies in the set.



Facts:

- If f is convex, then $\{x : f(x) \le 0\}$ is convex.
- The intersection of convex sets is convex.
  - If S_1 and S_2 are convex, then $S_2 := \{x \,:\, x \in S_1,\ x \in S_2\}$ is convex.

Descent Algorithms (Why Convex Optimization is Easy)

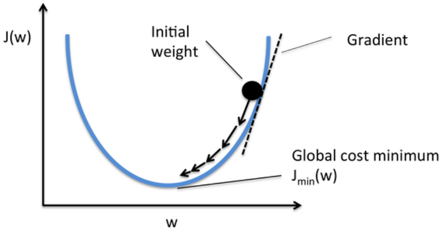
Unconstrained Optimization

All descent algorithms are iterative, with a search direction ($\Delta x \in \mathbb{R}^n$) and step size ($t \geq 0$).

$$x_{k+1} = x_k + t\Delta x$$

Gradient Descent

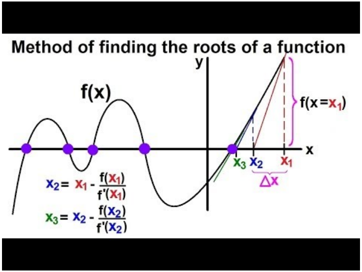
$$\Delta x = -\nabla f(x)$$



Newton's Algorithm:

$$\Delta x = -(\nabla^2 f(x))^{-1} \nabla f(x)$$

Tries to solve the equation $\nabla f(x) = 0$.



Both converge for sufficiently small step size.

Descent Algorithms (Why Convex Optimization is Easy)

Unconstrained Optimization

All descent algorithms are iterative, with a search direction ($\Delta x \in \mathbb{R}^n$) and step size ($t \geq 0$).

$$x_{k+1} = x_k + t\Delta x$$

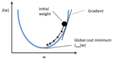
Gradient Descent

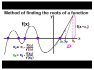
$$\Delta x = -\nabla f(x)$$

Newton's Algorithm:

$$\Delta x = -(\nabla^2 f(x))^{-1} \nabla f(x)$$

Tries to solve the equation $\nabla f(x) = 0$.





Both converge for sufficiently small step size.

- If $\nabla f(x) = 0$, then f has a minimum or maximum at x.
- In unconstrained optimization, the solution will occur at this inflection point.
- For a convex function, there is only one point where $\nabla f(x) = 0$, which is the global minimum.

Method 1: Gradient Projection

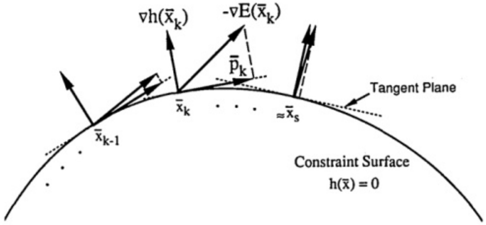


Figure: Must project step ($t\Delta x$) onto feasible Set

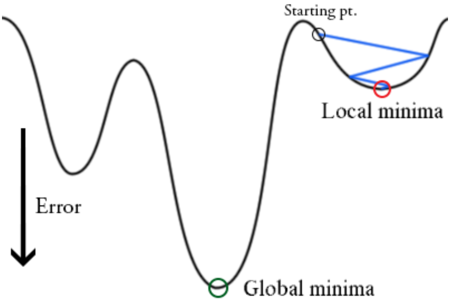
Method 2: Barrier Functions

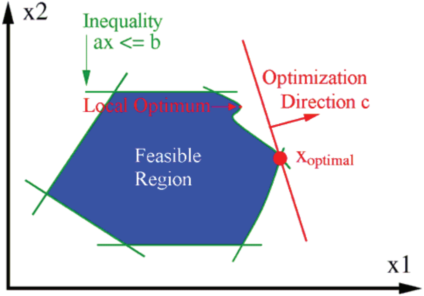
$$\min_x f(x) + \log(g(x))$$

Converts a Constrained problem to an unconstrained problem.
(Interior-Point Methods)

Non-Convexity and Local Optima

1. For convex optimization problems, Descent Methods always find the global optimal point.
2. For non-convex optimization, Descent Algorithms may get stuck at local optima.





Important Classes of Optimization Problems

Linear Programming

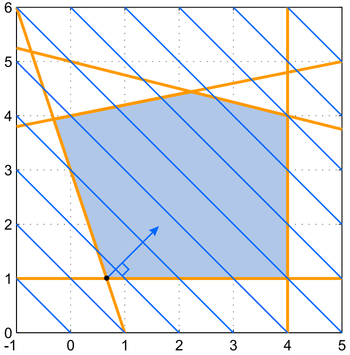
Linear Programming (LP)

$$\min_{x \in \mathbb{R}^n} c^T x : \qquad \text{subject to}$$

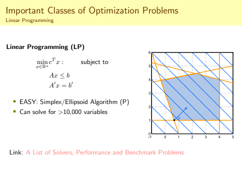
$$Ax \le b$$

$$A'x = b'$$

- EASY: Simplex/Ellipsoid Algorithm (P)
- Can solve for $>$10,000 variables



Link: A List of Solvers, Performance and Benchmark Problems



- The Ellipsoidal algorithm solves LP in polynomial time.
- The Simplex algorithm is not actually worst-case polynomial time.
- However, the simplex algorithm outperforms the ellipsoidal algorithm in almost all cases.

Quadratic Programming (QP)

$$\min_{x \in \mathbb{R}^n} x^T Q x + c^T x : \qquad \text{subject to}$$

$$Ax \leq b$$

- EASY (P): If $Q \geq 0$.
- HARD (NP-Hard): Otherwise

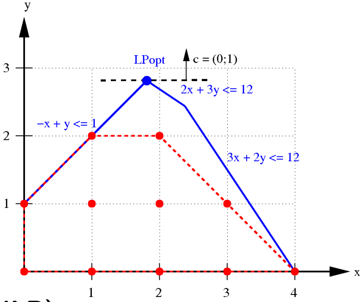
Mixed-Integer Linear Programming (MILP)

$$\min_{x\in\mathbb{R}^n} c^T x : \qquad \text{subject to}$$

$$Ax \le b$$

$$x_i \in \mathbb{Z} \qquad i = 1, \cdots K$$

- HARD (NP-Hard)



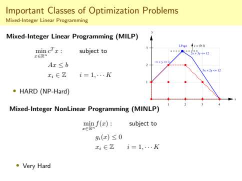
Mixed-Integer NonLinear Programming (MINLP)

$$\min_{x\in\mathbb{R}^n} f(x) : \qquad \text{subject to}$$

$$g_i(x) \le 0$$

$$x_i \in \mathbb{Z} \qquad i = 1, \cdots K$$

- Very Hard



- CPLEX and Gurobi will allow you to solve very large MILPs.
- However, the result may not be truly optimal.

Next Time:

Positive Matrices, SDP and LMIs

- Also a bit on Duality, Relaxations.