Convex Optimization & Optimization Tools

AA 203 Recitation #1

April 9th, 2021

Agenda

Preliminaries

- Why study Convex Optimization?
- Convex Sets & Convex Functions
- Convex Programming

Examples of Convex Optimization

- Linear Programming and Duality
- Quadratic Programming

CVXPY: Convex Optimization in Python

- Least Squares
- Discrete LQR

Preliminaries

Optimization

Optimization problems typically take the following form:

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } x \in S, \end{aligned}$$

where $f : S \to \mathbb{R}$ is a function and S is some some set that can generally be described by the intersection of equality and inequality constraints

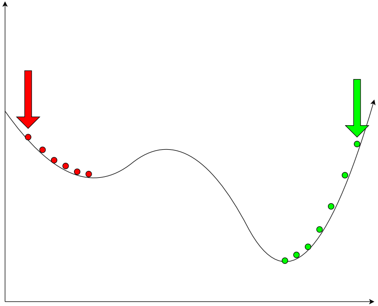
$$\begin{aligned} g_i(x) &\leq 0, \ \text{ for } i = 1, \ldots, m, \\ h_j(x) &= 0, \ \text{ for } j = 1, \ldots, k. \end{aligned}$$

Convex Optimization imposes a special structure of "convexity" on both the function f and the constraint set S

Why study Convex Optimization?

Observation 1: For convex optimization problems, every locally optimal solution is also globally optimal, i.e., every first order KKT solution is a global optimizer.

Observation 2: This is significant because numerical optimization algorithms like Gradient method and Newton Method can find first order KKT solutions/local minima.



Observation 3: Under non-convexities it is often computationally hard to find global minimizers.

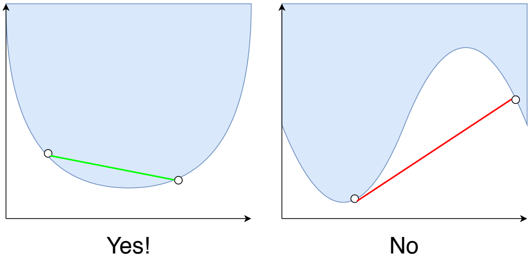
Convex Functions

Definition (Convex Functions)

A function $f : S \to \mathbb{R}$ is convex if for any $x_1, x_2 \in S$ and any $\alpha \in [0,1]$, it holds that

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2).$$

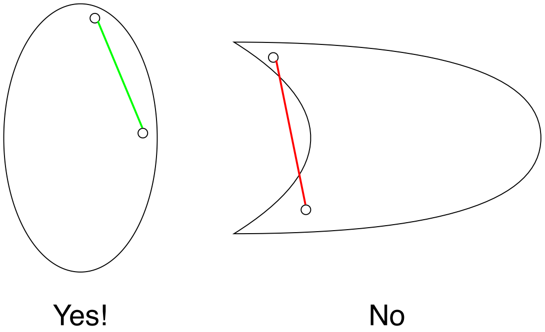
That is, a function is convex if the chord between $f(x_1)$ and $f(x_2)$ overestimates f between x_1 and x_2. Examples:



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:



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.

Suppose a set S is described by the intersection of equality and inequality constraints

$$g_i(x) \leq 0, \text{ for } i = 1, \ldots, m,$$
$$h_j(x) = 0, \text{ for } j = 1, \ldots, k.$$

Then, S is convex if the functions $h_j(x)$ are linear, and the functions $g_i(x)$ are convex.

Recipe to Identify Convex Programs

An optimization problem

$$\begin{aligned} \text{minimize } & f(x) \\ \text{subject to } & g_i(x) \leq 0, \ \text{for } i = 1, \ldots, m, \\ & h_j(x) = 0, \ \text{for } j = 1, \ldots, k. \end{aligned}$$

is convex if

1. The function $f(x)$ is convex
2. The functions $h_j(x)$ are linear
3. The functions $g_i(x)$ are convex

Examples

Is the following problem convex?

$$\begin{aligned} \text{minimize } & c^T x \\ \text{subject to } & a_i^T x \leq 0, \text{ for } i = 1, \dots, m, \\ & b_j^T x = 0, \text{ for } j = 1, \dots, k. \end{aligned}$$

This is a linear program - All linear programs are convex!
What about the following problem?

$$\begin{aligned} \text{minimize } & c^T x \\ \text{subject to } & ||x||^2 = 1. \end{aligned}$$

This problem is not convex, since the equality constraint is non-linear. But it can be convexified as:

$$\begin{aligned} \text{minimize } & c^T x \\ \text{subject to } & ||x||^2 \leq 1. \end{aligned}$$

Convex Program: Local Optima are Global Optima

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 $||x - x^*||_2 \leq \epsilon$, $f(x^*) \leq f(x)$.

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

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. $\square$

Convex Program: Local Optima are Global Optima

The result relies on both S, f being convex.

S not convex examples: Optimal Control of Nonlinear Systems, Integer Programming.

f not convex examples: Maximum Likelihood for Gaussian Mixtures, Training Neural Networks.

Examples of Convex Optimization

Optimization Models and Tools

We will focus on two of the most common convex Optimization Examples:

1. Linear Programming (LP) and Duality
2. Quadratic Programming (QP)

Other Common Optimization Models

- Semidefinite Programming (SDP).
- Convex Programming (CP).
- Mixed-Integer Linear Programming (IP).

Optimization Software

- CVXPY (LP, QP, SDP, CP, IP).
- CPLEX (LP, QP, IP).

Linear Programming

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

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

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 (CVXPY):

```
x = cvx.Variable(n)
prob = cvx.Problem(cvx.Minimize(c.T@x), [A @ x <= b])
prob.solve()
```

LP Duality

Suppose we have the following "Primal" linear program:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \ & c^T x \\ \text{subject to } & Ax \leq b, \\ & x \geq 0. \end{aligned}$$

Then, it has the following dual

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{maximize}} \ & b^T y \\ \text{subject to } & A^T y \geq -c, \\ & y \geq 0. \end{aligned}$$

Why is Duality Important?

Weak Duality: The optimal objective value of the dual problem is always a lower bound on the optimal objective value of the primal problem, i.e., $c^T x^* \geq b^T y^*$.

Strong Duality: If the primal problem has a feasible solution, then the optimal objective value of the dual problem is exactly equal to the optimal objective value of the primal problem, i.e., $c^T x^* = b^T y^*$.

Shadow Price Interpretation: The dual variables of the constraints of the primal problem can be interpreted as prices.

LP Example - Resource Allocation

Consider a scenario where m divisible resources $r_1, \ldots, r_m$ must be allocated to n people $t_1, \ldots, t_n$.

Each resource has a capacity of b_m units.

Each user can obtain at most one unit of resources

u_{ij} is the utility achieved when person p_i is allocated resource r_j.

Objective: Assign resources to people to maximize the total utility

LP Example - Resource Allocation

We can formulate the problem as a linear program with the decision variable: $x \in \mathbb{R}^{nm}$, where x_{ij} determines whether or not t_i is assigned resource r_j.

LP Example - Resource Allocation

We can formulate the problem as a linear program with the decision variable: $x \in \mathbb{R}^{nm}$, where x_{ij} determines whether or not t_i is assigned resource r_j.

LP Example - Resource Allocation

We can formulate the problem as a linear program with the decision variable: $x \in \mathbb{R}^{nm}$, where x_{ij} determines whether or not t_i is assigned resource r_j.

$$\underset{x \in \mathbb{R}^{nm}}{\text{maximize}} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{ij} x_{ij} \tag{1}$$

LP Example - Resource Allocation

We can formulate the problem as a linear program with the decision variable: $x \in \mathbb{R}^{nm}$, where x_{ij} determines whether or not t_i is assigned resource r_j.

$$\underset{x \in \mathbb{R}^{nm}}{\text{maximize}} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{ij} x_{ij} \tag{1}$$

$$\text{subject to } \sum_{i=1}^{n} x_{ij} \leq b_j \text{ for all } 1 \leq j \leq m \tag{2}$$

LP Example - Resource Allocation

We can formulate the problem as a linear program with the decision variable: $x \in \mathbb{R}^{nm}$, where x_{ij} determines whether or not t_i is assigned resource r_j.

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

LP Example - Resource Allocation

We can formulate the problem as a linear program with the decision variable: $x \in \mathbb{R}^{nm}$, where x_{ij} determines whether or not t_i is assigned resource r_j.

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

LP Example - Resource Allocation

We can formulate the problem as a linear program with the decision variable: $x \in \mathbb{R}^{nm}$, where x_{ij} determines whether or not t_i is assigned resource r_j.

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

(2) ensures that no good is sold more than its capacity. (3) ensures that no user gets more than one good.

LP Example - Resource Allocation

But how do we convince people that this is really the best allocation for them?

Let p be the prices in the market. Then, each person t_i wishes to maximize their payoff given by

$$\begin{aligned}\text{Payoff}_i &= \text{Total Utility accrued - Total Price Paid},\\ &= \sum_{j=1}^{m}(u_{ij} - p_j)x_{ij},\end{aligned}$$

subject to the constraint that they consume at most one resource.

That is, users wish to purchase any good j such that $j \in \arg\max_{j\in[m]}\{u_{ij} - p_j\}$ as long as $u_{ij} \geq p_j$ for some j.

LP Example - Resource Allocation

Let p_j be the dual of the capacity constraints and λ_i be the dual of the allocation constraints. Then, we have the following dual problem:

$$\begin{aligned} \underset{p\in\mathbb{R}^m,\lambda\in\mathbb{R}^n}{\text{minimize}} \ & \sum_{j=1}^{m} p_j b_j + \sum_{i=1}^{n} \lambda_i \\ \text{subject to } & \lambda_i \geq u_{ij} - p_j \text{ for all } 1 \leq i \leq n, 1 \leq j \leq m \\ & p \geq 0, \lambda \geq 0. \end{aligned}$$

The optimal solution is achieved when λ_i is minimized, i.e., $\lambda_i = \max_j\{u_{ij} - p_j\}$. Thus, the dual problem has the following economic interpretation:

1. p_j are the good prices
2. λ_i are agent utilities

LP Duality gives a method to set prices and achieve a decentralized implementation of the optimal solution.

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: Revenue Management, minimum weight matching, multi-commodity maximum flow, 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.

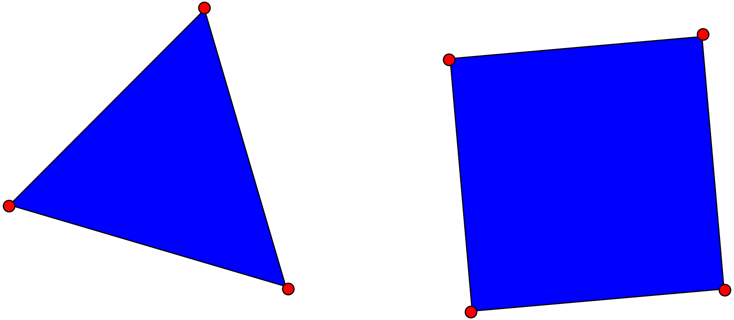
As a consequence, all points in S can be written as convex combinations of the extreme points of S.

Linear Programming - Properties

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

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*.*

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.

Quadratic Programming

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

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

where $H \succeq 0$, i.e., the matrix H is positive semi-definite.

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 (CVXPY):

```
x = cvx.Variable(n)
prob = cvx.Problem(cvx.Minimize((1/2) * cvx.quad_form(x, H) + f.T @ x), [A
@ x <= b, A_eq@x == b_eq])
prob.solve()
```

QP Example: Discrete LQR

Given a discrete linear dynamical system

$$x_{t+1} = Ax_t + Bu_t$$

The goal is to efficiently drive the state from x_0 to the origin. We incur a large cost if (a) the state is far from the origin or (b) we use a lot of control effort.

$$\frac{1}{2}x_T^\top Q_T x_T + \frac{1}{2}\sum_{t=0}^{T-1} x_t^\top Q x_t + u_t^\top R u_t$$

QP Example: Discrete LQR

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

$$
\begin{aligned}
\underset{u \in \mathbb{R}^T}{\text{minimize}} \quad & \frac{1}{2} x_T^\top Q_T x_T + \frac{1}{2} \sum_{t=0}^{T-1} x_t^\top Q x_t + u_t^\top R u_t \\
\text{subject to} \quad & x_{t+1} = A x_t + B u_t \text{ for all } 0 \leq t \leq T-1 && (4) \\
& x_0 = \text{initial condition} && (5) \\
& && (6)
\end{aligned}
$$

CVXPY: Convex Optimization in Python

Problem Objects in CVXPY

Instantiate by specifying an objective function and constraints.

```
prob = cvx.Problem(objective, constraints)
```

Specify a decision variable `x = cvx.Variable(n)`.

The `objective` is an expression, i.e. a function of the decision variable.

The `constraints` is a list of constraint objects.

Use `prob.solve()` to solve the problem.

Use `prob.status` to see if the optimization was successful.

The solution can then be found at `x.value`

The objective value of the solution can be found at `prob.value`

Least Squares in CVXPY

Recall the Least squares problem:

$$\min_{x \in \mathbb{R}^m} ||Ax - b||_2^2$$

where $A \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^n$.

Problem setup

```
import numpy as np
import cvxpy as cvx

n = 10
m = 5

A = np.random.normal(0,1,(n,m))
b = np.random.normal(0,1,(n,))
```

Least Squares in CVXPY

Solving the problem

```
x = cvx.Variable(m)

objective = cvx.Minimize(cvx.sum_squares(A @ x - b))
constraints = []

prob = cvx.Problem(objective, constraints)
prob.solve()

print(prob.status)
print(prob.value) # optimal objective value
print(x.value) # get the optimal solution
```

Discrete LQR in CVXPY

Recall the Discrete LQR problem:

$$
\begin{aligned}
\underset{u \in \mathbb{R}^T}{\text{minimize}} \quad & \frac{1}{2} x_T^\top Q_T x_T + \frac{1}{2} \sum_{t=0}^{T-1} x_t^\top Q x_t + u_t^\top R u_t \\
\text{subject to} \quad & x_{t+1} = A x_t + B u_t \text{ for all } 0 \leq t \leq T-1 \\
& x_0 = \text{initial condition}
\end{aligned}
$$

Discrete LQR in CVXPY

Problem setup

```
import numpy as np
import cvxpy as cvx

n = 5 # state dimension (x)
m = 5 # control dimenion (u)
T = 20 # number of timesteps in planning horizon
u_bound = 1.0 # bound on control effort

Q = np.eye(n) # state deviation cost
R = 2*np.eye(m) # control effort cost
A = np.random.normal(0,1,(n,n)) # dynamics
B = np.random.normal(0,1,(n,m))

x_0 = np.random.normal(0,1,(n,)) # initial condition
```

Discrete LQR in CVXPY

Iterative building of objective and constraints

```
X = {}
U = {}
cost_terms = []
constraints = []
```

Discrete LQR in CVXPY

Iterative building of objective and constraints

```
for t in range(T):
    X[t] = cvx.Variable(n) # state variable for time t
    U[t] = cvx.Variable(m) # control variable for time t
    cost_terms.append( cvx.quad_form(X[t],Q) ) # state cost
    cost_terms.append( cvx.quad_form(U[t],R) ) # control cost

    if (t == 0):
        constraints.append( X[t] == x_0) # initial condition

    if (t < T-1 and t > 0):
        # dynamics constraint
        constraints.append( A @ X[t-1] + B @ U[t-1] == X[t] )
```

Discrete LQR in CVXPY

Solving the Problem

```
objective = cvx.Minimize(cvx.sum(cost_terms))

prob = cvx.Problem(objective, constraints)
prob.solve()
print(prob.status) # optimal, infeasible, etc.
print(prob.value) # optimal objective value
print(U[0].value) # optimal control
```

Key Takeaways

1. Why it is important to study Convex Optimization
2. Basics of Convex Programming
3. Identifying Convex Programs
4. Basics of Linear Programming
5. Shadow Prices
6. Quadratic Programming
7. Basic Implementation on CVXPY