CVXPY x NASA Course 2024

Philipp Schiele Steven Diamond Parth Nobel Akshay Agrawal

June 10, 2024

Introduction

Outline

Course overview

Mathematical optimization

Convex optimization

Course overview

- weekly lectures Mondays 11:30–13:30 ET on Teams
- lectures include hands-on exercises for participants

- topics covered

June 10	Introduction to convex optimization and CVXPY
June 17	Disciplined convex programming
June 24	Landing a rocket using model predictive control
July 1	Sensitivity analysis and robust Kalman filtering
July 8	Regression and generalized linear models
July 15	Aircraft design using geometric programming
July 22	Code generation for quadcopter control
July 29	Object oriented optimization for power systems

Speakers

Philipp Schiele

- Completed PhD defense in statistics at LMU Munich (degree pending conferral)
- Postdoc at Stanford

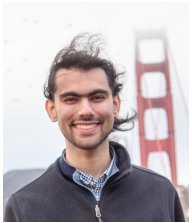
Steven Diamond

- PhD in computer science from Stanford
- Research scientist at Gridmatic

Speakers

Parth Nobel

- PhD candidate in electrical engineering at Stanford
- redesigned convex optimization course at Stanford



Akshay Agrawal

- PhD in electrical engineering from Stanford
- working on marimo

Outline

Course overview

Mathematical optimization

Convex optimization

Optimization problem

$$
\begin{array}{ll}
\text{minimize} & f_0(x) \\
\text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\
& g_i(x) = 0, \quad i = 1, \ldots, p
\end{array}
$$

- $x \in \mathbf{R}^n$ is (vector) variable to be chosen (n scalar variables $x_1, \ldots, x_n$)
- f_0 is the **objective function**, to be minimized
- $f_1, \ldots, f_m$ are the **inequality constraint functions**
- $g_1, \ldots, g_p$ are the **equality constraint functions**

- variations: maximize objective, multiple objectives, ...

Finding good (or best) actions

- x represents some **action**, *e.g.*,
  - trades in a portfolio
  - airplane control surface deflections
  - schedule or assignment
  - resource allocation
- constraints limit actions or impose conditions on outcome
- the smaller the objective $f_0(x)$, the better
  - total cost (or negative profit)
  - deviation from desired or target outcome
  - risk
  - fuel use

Finding good models

- x represents the **parameters** in a model
- constraints impose requirements on model parameters (*e.g.*, nonnegativity)
- objective $f_0(x)$ is sum of two terms:
  - a prediction error (or loss) on some observed data
  - a (regularization) term that penalizes model complexity

Worst-case analysis (pessimization)

- variables are actions or parameters out of our control (and possibly under the control of an adversary)
- constraints limit the possible values of the parameters
- minimizing $-f_0(x)$ finds **worst possible parameter values**

- if the worst possible value of $f_0(x)$ is tolerable, you're OK
- it's good to know what the worst possible scenario can be

Optimization-based models

- model an entity as taking actions that solve an optimization problem
  - an individual makes choices that maximize expected utility
  - an organism acts to maximize its reproductive success
  - reaction rates in a cell maximize growth
  - currents in a circuit minimize total power

- (except the last) these are **very crude** models
- and yet, they often work very well

Basic use model for mathematical optimization

- instead of saying how to choose (action, model)
- you articulate what you want (by stating the problem)
- then let an algorithm decide on (action, model)

Can you solve it?

- generally, no
- but you can try to solve it approximately, and it often doesn't matter

- the exception: **convex optimization**
  - includes linear programming (LP), quadratic programming (QP), many others
  - we can solve these problems reliably and efficiently
  - come up in many applications across many fields

Nonlinear optimization

traditional techniques for general nonconvex problems involve compromises

local optimization methods (nonlinear programming)

- find a point that minimizes f_0 among feasible points near it
- can handle large problems, *e.g.*, neural network training
- require initial guess, and often, algorithm parameter tuning
- provide no information about how suboptimal the point found is

global optimization methods

- find the (global) solution
- worst-case complexity grows exponentially with problem size
- often based on solving convex subproblems

Outline

Course overview

Mathematical optimization

Convex optimization

Convex optimization

convex optimization problem:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \end{array}$$

- variable $x \in \mathbf{R}^n$
- equality constraints are linear
- $f_0, \ldots, f_m$ are **convex**: for $\theta \in [0, 1]$,

$$f_i(\theta x + (1 - \theta)y) \leq \theta f_i(x) + (1 - \theta) f_i(y)$$

  i.e., f_i have nonnegative (upward) curvature

When is an optimization problem hard to solve?

- classical view:
  - linear (zero curvature) is easy
  - nonlinear (nonzero curvature) is hard
- the classical view is **wrong**
- the correct view:
  - convex (nonnegative curvature) is easy
  - nonconvex (negative curvature) is hard

Solving convex optimization problems

- many different algorithms (that run on many platforms)
  - interior-point methods for up to 10000s of variables
  - first-order methods for larger problems
  - do not require initial point, babysitting, or tuning
- can develop and deploy quickly using modeling languages such as CVXPY
- solvers are reliable, so can be embedded
- code generation yields real-time solvers that execute in milliseconds (*e.g.*, on Falcon 9 and Heavy for landing)

Modeling languages for convex optimization

- domain specific languages (DSLs) for convex optimization
  - describe problem in high level language, close to the math
  - can automatically transform problem to standard form, then solve

- enables rapid prototyping
- it's now much easier to develop an optimization-based application
- ideal for teaching and research (can do a lot with short scripts)

- gets close to the basic idea: **say what you want, not how to get it**

CVXPY example: non-negative least squares

math:

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|_2^2 \\ \text{subject to} & x \geq 0 \end{array}$$

- variable is x
- A, b given
- $x \geq 0$ means $x_1 \geq 0, \ldots, x_n \geq 0$

CVXPY code:

```
import cvxpy as cp

A, b = ...

x = cp.Variable(n)
obj = cp.norm2(A @ x - b)**2
constr = [x >= 0]
prob = cp.Problem(cp.Minimize(obj), constr)
prob.solve()
```

A world of optimization modeling languages

- express optimization problem in high level syntax
  - declare variables
  - form constraints and objective
  - solve
- long history: AMPL, GAMS, ...
  - no special support for convex problems
  - very limited syntax
  - callable from, but not embedded in other languages
- DCP-based modeling: YALMIP, CVX, Convex.jl, **CVXPY**

Course goals

Abstractly, the goal of this course is to enable you to

1. recognize problems that can be formulated as convex optimization problems
2. if necessary, reformulate problems by applying transformations
3. specify problems in CVXPY
4. solve problems fast and reliably

Brief history of convex optimization

- **theory (convex analysis)**: 1900–1970
- **algorithms**
  - 1947: simplex algorithm for linear programming (Dantzig)
  - 1960s: early interior-point methods (Fiacco & McCormick, Dikin, ...)
  - 1970s: ellipsoid method and other subgradient methods
  - 1980s & 90s: interior-point methods (Karmarkar, Nesterov & Nemirovski)
  - since 2000s: many methods for large-scale convex optimization
- **applications**
  - before 1990: mostly in operations research, a few in engineering
  - since 1990: many applications in engineering (control, signal processing, communications, circuit design, ...)
  - since 2000s: machine learning and statistics, finance

Summary

convex optimization problems

- are optimization problems of a special form
- arise in many applications
- can be solved effectively
- are easy to specify using DSLs

Disciplined Convex Programming

Outline

Convex Optimization

Constructive Convex Analysis

Disciplined Convex Programming (DCP)

Convex optimization problem — standard form

$$
\begin{array}{ll}
\text{minimize} & f_0(x) \\
\text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\
& Ax = b
\end{array}
$$

with variable $x \in \mathbf{R}^n$

- objective and inequality constraints $f_0, \ldots, f_m$ are convex

  for all $x, y, \theta \in [0, 1]$,

  $$f_i(\theta x + (1 - \theta)y) \leq \theta f_i(x) + (1 - \theta) f_i(y)$$

  i.e., graphs of f_i curve upward
- equality constraints are linear

Convex optimization problem — conic form

cone program:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b, \quad x \in \mathcal{K} \end{array}$$

with variable $x \in \mathbf{R}^n$

- linear objective, equality constraints; $\mathcal{K}$ is convex cone
- special cases:
  - linear program (LP)
  - second-order cone program (SOCP)
  - semidefinite program (SDP)

- the modern canonical form
- *there are well developed solvers for cone programs*

How do you solve a convex problem?

- use an existing custom solver for your specific problem
- develop a new solver for your problem using a currently fashionable method
  - requires work
  - but (with luck) will scale to large problems
- transform your problem into a cone program, and use a standard cone program solver
  - can be *automated* using *domain specific languages*

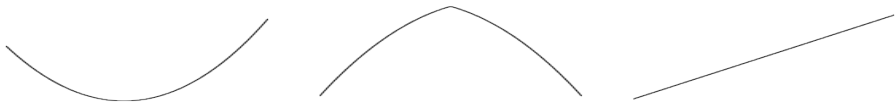
Outline

Convex Optimization

Constructive Convex Analysis

Disciplined Convex Programming (DCP)

Curvature: Convex, concave, and affine functions



- f is *concave* if $-f$ is convex, *i.e.*, for any $x, y, \theta \in [0, 1]$,

$$f(\theta x + (1-\theta)y) \geq \theta f(x) + (1-\theta)f(y)$$

- f is *affine* if it is convex and concave, *i.e.*,

$$f(\theta x + (1-\theta)y) = \theta f(x) + (1-\theta)f(y)$$

  for any $x, y, \theta \in [0, 1]$
- f is affine $\iff$ it has form $f(x) = a^T x + b$

Examples of basic convex functions

- x^p ($p \geq 1$ or $p \leq 0$), *e.g.*, x^2, $1/x$ ($x > 0$)
- e^x
- $x \log x$
- $a^T x + b$
- $x^T P x$ ($P \succeq 0$)
- $\|x\|$ (any norm)
- $\max(x_1, \ldots, x_n)$

Examples of basic concave functions

- x^p $(0 \leq p \leq 1)$, *e.g.*, $\sqrt{x}$
- $\log x$
- $\sqrt{xy}$
- $x^T P x$ $(P \preceq 0)$
- $\min(x_1, \ldots, x_n)$

Less basic examples

Convex functions

- x^2/y for $y > 0$ and $x^T Y^{-1} x$ for $Y \succ 0$.
- $x \log(x/y)$ for $x, y \geq 0$.
- $\lambda_{\max}(X)$, for symmetric X.

Concave functions

- $\log \det X$ and $(\det X)^{1/n}$, for $X \succ 0$.
- $\log \Phi(x)$, where Φ is the Gaussian CDF.
- $\lambda_{\min}(X)$, for symmetric X.

How to verify that a function is convex or concave?

- via the definition. For convex functions,

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$$

- via first or second order conditions. For convex functions,

$$\nabla^2 f(x) \succeq 0$$

- via convex calculus:
  - start w/ library of basic functions that are convex or concave
  - apply transformations that preserve convexity

Convex calculus: basic rules

- **nonnegative scaling**: f convex, $\alpha \geq 0 \implies \alpha f$ convex
- **sum**: f, g convex $\implies f + g$ convex
- **affine composition**: f convex $\implies f(Ax + b)$ convex
- **pointwise maximum**: $f_1, \ldots, f_m$ convex $\implies \max_i f_i(x)$ convex
- **composition**: h convex increasing, f convex $\implies h(f(x))$ convex

. . . and similar rules for concave functions

Convex calculus: applications

- ℓ_1-regularized least-squares cost:

$$\|Ax - b\|_2^2 + \lambda \|x\|_1, \text{ with } \lambda \geq 0$$

- sum of largest k elements of x:

$$x_{[1]} + \cdots + x_{[k]}$$

- log-barrier:

$$\sum_{i=1}^{m} \log(-f_i(x)) \quad \text{on} \quad \{x \mid f_i(x) < 0\}, f_i \text{ convex.}$$

One Rule to Rule Them All

$h(f_1(x), \ldots, f_k(x))$ is convex when h is convex and for each i

- h is increasing in argument i, and f_i is convex, or
- h is decreasing in argument i, and f_i is concave, or
- f_i is affine

There's a similar rule for concave compositions
(just swap convex and concave above).

Example: The One Rule

let's show that

$$f(u, v) = (u + 1) \log\left(\frac{u + 1}{\min(u, v)}\right)$$

is convex

three steps:

1. $x \log(x/y)$ is convex in (x, y), decreasing in y
2. u, v are variables with $u, v > 0$
3. $u + 1$ is affine; $\min(u, v)$ is concave; both positive

Outline

Convex Optimization

Constructive Convex Analysis

Disciplined Convex Programming (DCP)

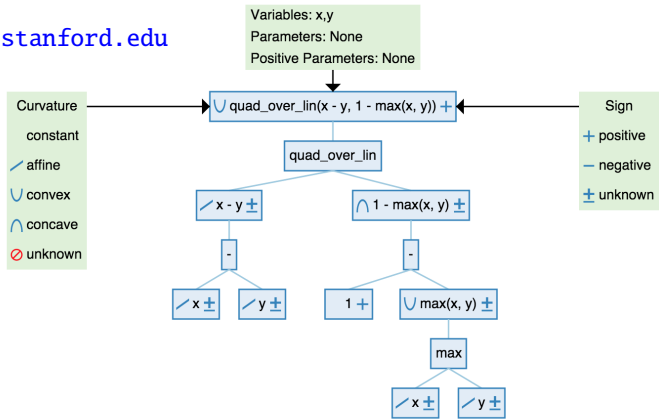
Algorithmic convexity verification: idea

- start with an **Expression**, build a parse tree
  - leaves: variables, constants, or parameters
  - nodes: **Atom** objects (functions of children)
- store curvature + monotonicity info of leaves and nodes
  - convex, concave, affine, constant
  - increasing, decreasing
  - helps to tag *signs*. E.g. x^2 increasing for $x \geq 0$
- apply The One Rule from the bottom up

Algorithmic convexity verification: example

$$(x-y)^2/(1-\max(x,y))$$

dcp.stanford.edu



Exercise: DCP analysis

https://marimo.app/l/3s4nd6

Solution: DCP analysis

https://marimo.app/l/bi9huq

Exercise: limitations of CVXPY's DCP parsing

Your task:

- find a convex CVXPY Expression `expr` for which `expr.is_dcp() == False`

recall: $h(f_1(x), \ldots, f_k(x))$ is convex when h is convex and for each i

- h is increasing in argument i, and f_i is convex, or
- h is decreasing in argument i, and f_i is concave, or
- f_i is affine

Exercise: limitations of CVXPY's DCP parsing

$\sqrt{x^2 + 1}$

- bad: `cp.sqrt(cp.square(x) + 1)`
  - leaves: 1 is constant, x is affine
  - node: x^2 is convex
  - node: $1 + x^2$ is convex
  - node: $\sqrt{\cdot}$ is concave!

- good: `cp.norm(cp.hstack([x, 1]), 2)`
  - $x \mapsto [x, 1]$ is affine
  - $\|\cdot\|_2$ is convex

Backup slides

Exercise: Python intro

https://marimo.app/l/14bxi2