CVXPY x NASA Course 2024

Philipp Schiele   Steven Diamond   Parth Nobel   Akshay Agrawal

July 29, 2024

CVXPY

Nonconvex Optimization Examples

Outline

HW 1 Solution

`https://marimo.app/l/6zclft`

HW 2 Solution

`https://marimo.app/l/cxfkhn`

This lecture

- nonlinear programming and the DCCP package
- integer programming
- solvers available for CVXPY

Resources

- Disciplined Convex-Concave Programming [Shen et al.]
- `https://github.com/cvxgrp/dccp`
- Mosek Modeling Cookbook
- `https://docs.mosek.com/modeling-cookbook/mio.html`

Outline

Homework review

Nonlinear programming

Mixed-integer programming

Solvers

Conclusion

Nonlinear program (NLP)

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

- variable $x \in \mathbf{R}^n$
- $f_0, \ldots, f_m$ and $g_1, \ldots, g_p$ are differentiable with gradient oracles
- ideally twice differentiable with Hessian oracles

NLP pros

- (twice) differentiability is a simple requirement
- can build on autodifferentiation tools
- good fit for nonlinear dynamics
- good fit for advanced integration schemes

NLP cons

- no guarantees for arbitrary problems
- often requires baby sitting to get good results
- results highly dependent on algorithm details

IPOPT

- the most commonly used solver for nonlinear programs
- open-source
- developed for over 20 years
- interfaces in many languages
- `https://coin-or.github.io/Ipopt/`

CasADi

- a framework for nonlinear programming
- sophisticated autodifferentiation
- widely used for control applications
- code generation capabilities
- supports many different solvers
- `https://web.casadi.org/`

DCCP

- a CVXPY extension for certain types of NLPs
- limited parameter tuning (no trust region)
- guaranteed to improve every iteration
- can still get stuck at local optima
- `https://github.com/cvxgrp/dccp`

DCCP

DCCP standard form

$$\begin{array}{ll} \text{minimize/maximize} & o(x) \\ \text{subject to} & l_i(x) \sim r_i(x), \quad i = 1, \ldots, m \end{array}$$

- variable $x \in \mathbf{R}^n$
- o, $l_1, \ldots, l_m$, and $r_1, \ldots, r_m$ are DCP
- $\sim$ could mean $=$, $\leq$, or $\geq$

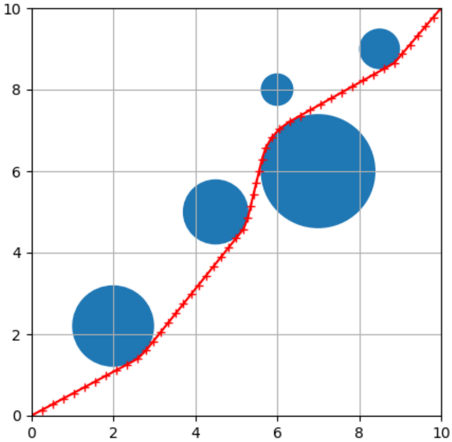
Obstacle avoidance

- simulated in 2D
- move from point (0,0) to point (10,10)
- take small steps
- avoid circular obstacles

Obstacle avoidance

```
import dccp
X = cp.Variable((d, T+1))
L = cp.Variable()
constr = [
    X[:, 0] == a,
    X[:, T] == b,
]
for t in range(T):
    # Take small steps.
    constr += [cp.norm(X[:, t] - X[:, t+1], 2) <= L / T]
    for j in range(m):
        # Avoid circular obstacles.
        constr += [cp.norm(X[:, t] - p[:, j], 2) >= r[j]]
prob = cp.Problem(cp.Minimize(cost), constr)
result = prob.solve(method="dccp")
```

Obstacle avoidance



Outline

Mixed-integer program (MIP)

- requires that certain variables be integers or booleans (i.e., 0 or 1)
- historically focused on linear programs with integer/boolean variables
- current MIP solvers can also solve quadratic and second-order cone programs

MIP example

- choose items to maximize value while respecting weight limit
  - items: A, B, C
  - values: \$6, \$5, \$4
  - weights: 2 kg, 3 kg, 1 kg
  - knapsack capacity: 5 kg
- decision variables: $x_A, x_B, x_C \in \{0, 1\}$

MIP example

$$\begin{array}{ll} \text{maximize} & 6x_A + 5x_B + 4x_C \\ \text{subject to} & 2x_A + 3x_B + x_C \leq 5 \\ & x_A, x_B, x_C \in \{0, 1\} \end{array}$$

where:

- $x_i = 1$ if item i is selected, 0 otherwise
- objective: maximize total value
- constraint: total weight $\leq$ knapsack capacity

MIP example

- solution: $x_A = 1, x_B = 1, x_C = 0$
- interpretation:
  - select items A and B
  - total value: \$11
  - total weight: 5 kg

MIP pros

- can model enormous class of problems (NP-hard)
- natural when making discrete decisions
- often can be solved efficiently in practice

MIP cons

- best results require closed-source black magic (Gurobi)
- problems may be very slow to solve
- (in my opinion) leads to out of control complexity

Exercise

`https://marimo.app/l/j8o7av`

Solution

`https://marimo.app/l/okmcmg`

Outline

Solvers

- CVXPY has interfaces to 21 different solvers
- if you don't specify a solver, CVXPY chooses for you
- based on the problem type and an internal solver ordering

Built-in solvers

- CVXPY comes with four open-source solvers
- Clarabel
  - the default solver for most problems
  - handles most problem types
- OSQP
  - the default for quadratic programs
  - very fast
- SCS
  - the default for semidefinite programs
  - better for large problems
- ECOS (deprecated)
  - the default solver for many years

Other open-source solvers

- can use SciPy >= 1.9 for mixed-integer linear programs
- many interesting alternatives for
  - mixed-integer programs
  - linear programs
  - small/dense quadratic programs
  - ultra high-precision
- if your solution looks wrong, print the solver output
- try more than one solver

Commercial solvers

- Mosek
  - excellent for convex problems
  - fast and reliable
  - handles most problem types
- Gurobi
  - excellent for mixed-integer programs
  - expensive
- many other great choices

Outline

Conclusion

- convex optimization problems arise in many applications
  - rocket landing
  - robust Kalman filter
  - statistical estimation
  - aircraft design
  - ...
- CVXPY makes these problems easy to model and solve

Conclusion

- you can turn a CVXPY model into C code with CVXPYgen
- convex optimization is not the only tool
  - nonlinear programming
  - mixed-integer programming
  - ...
- join us on the CVXPY discord! (`https://discord.gg/4urRQeGBCr`)