Topics in Optimization and its Applications to Computer Science: CVXPY

Chee Wei TAN

Introduction

CVXPY is a Python-embedded modeling language for convex optimization problems.

You take the driver seat expressing your problem in a natural way that follows the math, rather than in a restrictive standard form required by solvers.

CVXPY is part of an ecosystem of optimization software that adheres to *Disciplined Convex Programming* (DCP) developed by Stephen Boyd's group at Stanford.

`https://www.cvxpy.org/index.html`

Example 1: Least-squares

$$\underset{x}{\text{minimize}} \quad ||Ax - b||_2^2$$

Here $A \in \mathcal{R}^{m \times n}$ and $b \in \mathcal{R}^m$ are problem data and $x \in \mathcal{R}^n$ is the optimization variable.

```python
import cvxpy as cp
import numpy as np

# Generate data.
m = 20
n = 15
np.random.seed(1)
A = np.random.randn(m, n)
b = np.random.randn(m)

# Define and solve the CVXPY problem.
x = cp.Variable(n)
cost = cp.sum_squares(A*x - b)
prob = cp.Problem(cp.Minimize(cost))
prob.solve()

# Print result.
print("\nThe optimal value is", prob.value)
print("The optimal x is")
print(x.value)
print("The norm of the residual is ", cp.norm(A*x - b, p=2).value)
```

Example 2: Constrained Least Squares

The following code solves a least-squares problem with box constraints:

```
import cvxpy as cp
import numpy as np

# Problem data.
m = 30
n = 20
np.random.seed(1)
A = np.random.randn(m, n)
b = np.random.randn(m)

# Construct the problem.
x = cp.Variable(n)
objective = cp.Minimize(cp.sum_squares(A*x - b))
constraints = [0 <= x, x <= 1]
prob = cp.Problem(objective, constraints)

# The optimal objective value is returned by `prob.solve()`.
result = prob.solve()
# The optimal value for x is stored in `x.value`.
print(x.value)
# The optimal Lagrange multiplier for a constraint is stored in
# `constraint.dual_value`.
print(constraints[0].dual_value)
```

Example 3: Linear Program

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^T x \\ \text{subject to} & Ax \leq b \end{array}$$

Here $A \in \mathcal{R}^{m \times n}$, $b \in \mathcal{R}^m$, and $c \in \mathcal{R}^n$ are problem data and $x \in \mathcal{R}^n$ is the optimization variable.

```python
# Import packages.
import cvxpy as cp
import numpy as np

# Generate a random non-trivial linear program.
m = 15
n = 10
np.random.seed(1)
s0 = np.random.randn(m)
lamb0 = np.maximum(-s0, 0)
s0 = np.maximum(s0, 0)
x0 = np.random.randn(n)
A = np.random.randn(m, n)
b = A@x0 + s0
c = -A.T@lamb0

# Define and solve the CVXPY problem.
x = cp.Variable(n)
prob = cp.Problem(cp.Minimize(c.T@x),
                 [A@x <= b])
prob.solve()

# Print result.
print("\nThe optimal value is", prob.value)
print("A solution x is")
print(x.value)
print("A dual solution is")
print(prob.constraints[0].dual_value)
```

Example 4: Quadratic Program

$$\begin{array}{ll} \underset{x}{\text{minimize}} & (1/2)x^T P x + q^T x \\ \text{subject to} & Gx \leq h \\ & Ax = b \end{array}$$

Here $P \in \mathcal{S}^n_+$, $q \in \mathcal{R}^n$, $G \in \mathcal{R}^{m \times n}$, $h \in \mathcal{R}^m$, $A \in \mathcal{R}^{p \times n}$, and $b \in \mathcal{R}^p$ are problem data and $x \in \mathcal{R}^n$ is the optimization variable.

```python
import cvxpy as cp
import numpy as np

# Generate a random non-trivial quadratic program.
m = 15
n = 10
p = 5
np.random.seed(1)
P = np.random.randn(n, n)
P = P.T@P
q = np.random.randn(n)
G = np.random.randn(m, n)
h = G@np.random.randn(n)
A = np.random.randn(p, n)
b = np.random.randn(p)

# Define and solve the CVXPY problem.
x = cp.Variable(n)
prob = cp.Problem(cp.Minimize((1/2)*cp.quad_form(x, P) + q.T@x),
                 [G@x <= h,
                  A@x == b])
prob.solve()

# Print result.
print("\nThe optimal value is", prob.value)
print("A solution x is")
print(x.value)
print("A dual solution corresponding to the inequality constraints is")
print(prob.constraints[0].dual_value)
```