Disciplined Convex Stochastic Programming: A New Framework for Stochastic Optimization

Supplementary Material

In Sec. 1 of this supplement, we continue Sec. 4 of our paper and present another example of a stochastic program along with its corresponding `cvxstoc` implementation. In Sec. 2, we give a PySP [2] implementation of the news vendor problem that was outlined in Sec. 4.2 of the paper.

1 NETWORK RESOURCE ALLOCATION

Consider the (general) problem of allocating resources across a network; we make this problem concrete by focusing on the specific task of (optimally) allocating airline passengers to flights, subject to flight capacity restrictions, in the face of uncertain demand [1, chap. 16].

To this end, suppose we have a graph $G = (\mathcal{V}, \mathcal{E})$, with a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$, and let $\mathcal{P}$ be a set of possible paths in the graph; we also define $P = |\mathcal{P}|$, $n = |\mathcal{V}|$, and $E = |\mathcal{E}|$. We wish to maximize revenue, which we earn on a per-path basis (some paths may be more lucrative than others) and model as a vector of prices $p \in \mathbf{R}^P$; however, demand for paths is uncertain and is therefore modeled as a random vector $d \in \mathbf{R}^P$ with $d_i \sim \mathrm{LogNormal}(\mu_i, \sigma_i^2),\ i = 1, \ldots, P$.

We assume the set of passengers is partitioned into a set of C passenger categories (*e.g.*, business traveler, government, consumer, *etc.*), and that the number of passengers in each category (which we denote as $x_i \in \mathbf{R}^E,\ i = 1, \ldots, C$) that we may assign to each edge is constrained by some vector $u \in \mathbf{R}^E$.

We can pose this problem as the following two-stage stochastic program:

$$\begin{array}{ll} \underset{x_1,\ldots,x_C}{\text{minimize}} & \mathbf{E}\, Q(x_1, \ldots, x_C) \\ \text{subject to} & \sum_{i=1}^C x_i \preceq u, \quad i = 1, \ldots, C \\ & x_i \succeq 0, \quad i = 1, \ldots, C, \end{array}$$

$$\text{where } Q(x_1, \ldots, x_C) = \begin{array}{ll} \underset{y}{\min} & -p^T y \\ \text{s.t.} & \sum_{k \in \mathcal{P}_i^j} y_k \leq (x_j)_i, \\ & i = 1, \ldots, C, \\ & j = 1, \ldots, E, \\ & 0 \preceq y \preceq d, \end{array} \quad (1)$$

and $\mathcal{P}_i^j$ denotes the set of paths containing edge j that are flown by category i (such information is assumed to be known a priori). Note that the first constraint in (1) merely enforces consistency between the x and y variables, and can also be written more compactly as $A_i y \preceq x_i,\ i = 1, \ldots, C$, for appropriately-defined matrices $A_i \in \mathbf{R}^{E \times P}$.

A `cvxstoc` implementation of this network resource allocation problem is given in Listing 1 (the problem data is the same as [1, chap. 16], *i.e.*, $n = 7$, $P = 6$, $E = 6$, $A_i \in \mathbf{R}^{6 \times 6}$, and $C = 2$).

```
# Create optimization variables
x = [NonNegative(E) for i in range(C)]
y = NonNegative(P)

# Create second stage problem
capacity = [A[i]*y<=x[i] for i in range(C)]
d = RandomVariable(pymc.Lognormal(name="d", mu=0,
                                   tau=1, size=P))
p2 = Problem(Minimize(-y.T*p), [y<=d] + capacity)
Q = partial_optimize(p2, [y], [x[0], x[1]])

# Create and solve first stage problem
p1 = Problem(Minimize(expectation(Q(*x), m)),
             [sum(x) <= u])
p1.solve()
```

Listing 1: A `cvxstoc` implementation of a network resource allocation problem.

2 A PySP IMPLEMENTATION OF THE NEWS VENDOR PROBLEM

In this section, we give a PySP [2] implementation of the news vendor problem described in Sec. 4.2 of our paper. Listing 2 specifies the first and second stage objective functions in PySP syntax, as well as declares the optimization variables, random variables, and problem data. Listing 3 specifies the relevant probability distributions, while Listing 4 describes the random variables, as well as the values of the problem data.

```
# Helper functions
def obj_rule(model):
    return model.FirstStageCost + model.SecondStageCost

def ComputeFirstStageCost_rule(model):
    return (model.FirstStageCost - (model.b*model.x))
     ↪ == 0

def ComputeSecondStageCost_rule(model):
    return (model.SecondStageCost - (-model.s*model.y1
     ↪ - model.r*model.y2)) == 0

def constr1_rule(model):
    return (model.y1+model.y2) <= model.x

def constr2_rule(model, i):
    return 0 <= model.y1 and model.y1 <= model.d[i]

# Initialize problem data
model = AbstractModel()

model.b = Param()

model.s = Param()
model.r = Param()

model.scens = Set()
model.d = Param(model.scens)

model.u = Param()

# Setup all stage problems
model.x = Var(bounds=(0.0, model.u))
model.y1 = Var()
model.y2 = Var(within=NonNegativeReals)

model.obj = Objective(rule=obj_rule, sense=minimize)
model.FirstStageCost = Var()
model.SecondStageCost = Var()
model.ComputeFirstStageCost = Constraint(rule=
     ↪ ComputeFirstStageCost_rule)
model.ComputeSecondStageCost = Constraint(rule=
     ↪ ComputeSecondStageCost_rule)

model.constr1 = Constraint(rule=constr1_rule)
model.constr2 = Constraint(model.scens, rule=
     ↪ constr2_rule)
```

Listing 2: A PySP implementation of the news vendor problem.

```
set Nodes := RootNode
             BelowAverageNode
             AverageNode
             AboveAverageNode;

param NodeStage := RootNode            FirstStage
                   BelowAverageNode    SecondStage
                   AverageNode         SecondStage
                   AboveAverageNode    SecondStage;

set Children[RootNode] := BelowAverageNode
                          AverageNode
                          AboveAverageNode;

param ConditionalProbability := RootNode         1.0
                                BelowAverageNode 0.3
                                AverageNode      0.6
                                AboveAverageNode 0.1;

set Scenarios := BelowAverageScenario
                 AverageScenario
                 AboveAverageScenario;

param ScenarioLeafNode :=
      BelowAverageScenario    BelowAverageNode
      AverageScenario         AverageNode
      AboveAverageScenario    AboveAverageNode;

set StageVariables[FirstStage] := x;
set StageVariables[SecondStage] := y1
                                   y2;

param StageCostVariable :=
      FirstStage     FirstStageCost
      SecondStage    SecondStageCost;
```

Listing 3: A PySP implementation of the news vendor problem.

```
param s := 25;
param r := 5;

set scens := BELOW AVG ABOVE;
param d := BELOW 55 AVG 139 ABOVE 141;

param u := 150;
```

Listing 4: A PySP implementation of the news vendor problem.

References

[1] S. Wallace and W. Ziemba. *Applications of Stochastic Programming*. MPS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, 2005.

[2] J. Watson, D. Woodruff, and W. Hart. PySP: Modeling and solving stochastic programs in Python. *Mathematical Programming Computation*, 4(2):109–149, 2012.