Radiation treatment planning

[1]:
```
import cvxpy as cp
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# Load data
n = 300
mtumor = 100
mother = 400
Atumor = np.loadtxt('Atumor.csv', delimiter=',')
Aother = np.loadtxt('Aother.csv', delimiter=',')
Bmax = 10
Dtarget = 1
Dother = 0.25
```

[3]:
```
tumor,n = Atumor.shape
other,n = Aother.shape
tumor,other,n
```

[3]:
```
(100, 400, 300)
```

So $\mathcal{T} = \{1, 2, \cdots, 100\}, n = 300, m = 500$. Recall that the doses were given by $d = Ab$.

Solving the original optimization problem

Here we solve the problem

$$\begin{aligned} \min \ & \sum_{i \notin \mathcal{T}} \max(d_i - D^{\text{other}}, 0) \\ \text{s.t. } & d = Ab \\ & d_i \geq D^{\text{target}} \ \forall i \in \mathcal{T} \\ & 0 \leq b_i \leq B^{\max} \ \forall 1 \leq i \leq n \end{aligned}$$

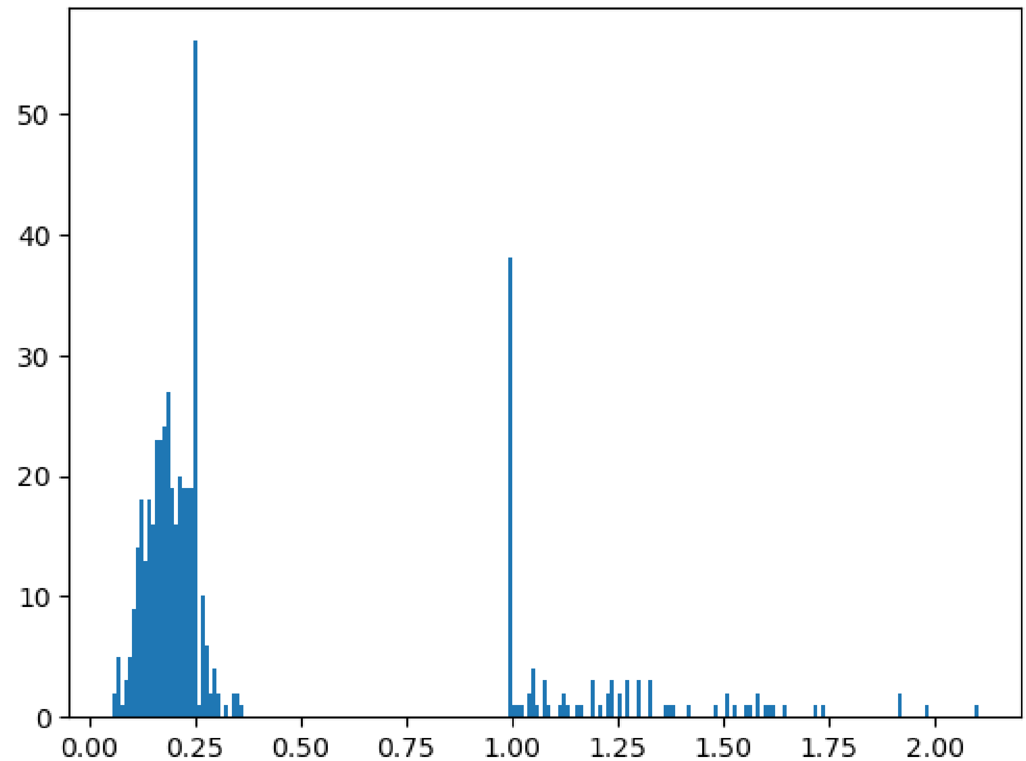
[4]:
```
dtumor = cp.Variable(tumor, 'dtumor')
dother = cp.Variable(other, 'dother')
b = cp.Variable(n, 'b')
f = cp.maximum(dother - Dother, 0)
obj = cp.sum(f)
```

```python
cons = [Aother @ b == dother, Atumor @ b == dtumor, dtumor >= Dtarget, b >= 0,␣
 ↪Bmax >= b]
problem = cp.Problem(cp.Minimize(obj), cons)
problem.solve(verbose = True, solver = cp.ECOS)
```

```
===============================================================================
                                     CVXPY
                                     v1.4.2
===============================================================================
(CVXPY) Mar 19 06:40:33 PM: Your problem has 800 variables, 5 constraints, and 0
parameters.
(CVXPY) Mar 19 06:40:33 PM: It is compliant with the following grammars: DCP,
DQCP
(CVXPY) Mar 19 06:40:33 PM: (If you need to solve this problem multiple times,
but with different data, consider using parameters.)
(CVXPY) Mar 19 06:40:33 PM: CVXPY will first compile your problem; then, it will
invoke a numerical solver to obtain a solution.
(CVXPY) Mar 19 06:40:33 PM: Your problem is compiled with the CPP
canonicalization backend.
-------------------------------------------------------------------------------
                                  Compilation
-------------------------------------------------------------------------------
(CVXPY) Mar 19 06:40:33 PM: Compiling problem (target solver=ECOS).
(CVXPY) Mar 19 06:40:33 PM: Reduction chain: Dcp2Cone -> CvxAttr2Constr ->
ConeMatrixStuffing -> ECOS
(CVXPY) Mar 19 06:40:33 PM: Applying reduction Dcp2Cone
(CVXPY) Mar 19 06:40:33 PM: Applying reduction CvxAttr2Constr
(CVXPY) Mar 19 06:40:33 PM: Applying reduction ConeMatrixStuffing
(CVXPY) Mar 19 06:40:33 PM: Applying reduction ECOS
(CVXPY) Mar 19 06:40:33 PM: Finished problem compilation (took 1.760e-02
seconds).
-------------------------------------------------------------------------------
                                Numerical solver
-------------------------------------------------------------------------------
(CVXPY) Mar 19 06:40:33 PM: Invoking solver ECOS  to obtain a solution.
-------------------------------------------------------------------------------
                                    Summary
-------------------------------------------------------------------------------
(CVXPY) Mar 19 06:40:33 PM: Problem status: optimal
(CVXPY) Mar 19 06:40:33 PM: Optimal value: 1.388e+00
(CVXPY) Mar 19 06:40:33 PM: Compilation took 1.760e-02 seconds
(CVXPY) Mar 19 06:40:33 PM: Solver (including time spent in interface) took
1.676e-01 seconds
```

[4]: 1.3882005424049697

[5]:
```python
#d = np.concatenate(np.array(dother.value), np.array(dtumor.value))
d = [0] * (tumor+other)
for i in range(tumor):
    d[i] = dtumor.value[i]
for i in range(other):
    d[i+tumor] = dother.value[i]
plt.hist(d, bins=225)
plt.xticks(np.arange(0, 2.25, 0.25))
plt.show()
```



Solving the (coverted) linear optimization problem

$$
\begin{aligned}
\min \ & \sum_{i \notin \mathcal{T}} u_i \\
\text{s.t. } & d = Ab \\
& d_i \geq D^{\text{target}} \ \forall i \in \mathcal{T} \\
& 0 \leq b_i \leq B^{\max} \ \forall 1 \leq i \leq n \\
& 0 \leq u_i \ \forall 1 \leq i \leq m, i \notin \mathcal{T} \\
& d_i - D^{\text{other}} \leq u_i \ \forall 1 \leq i \leq m, i \notin \mathcal{T}.
\end{aligned}
$$

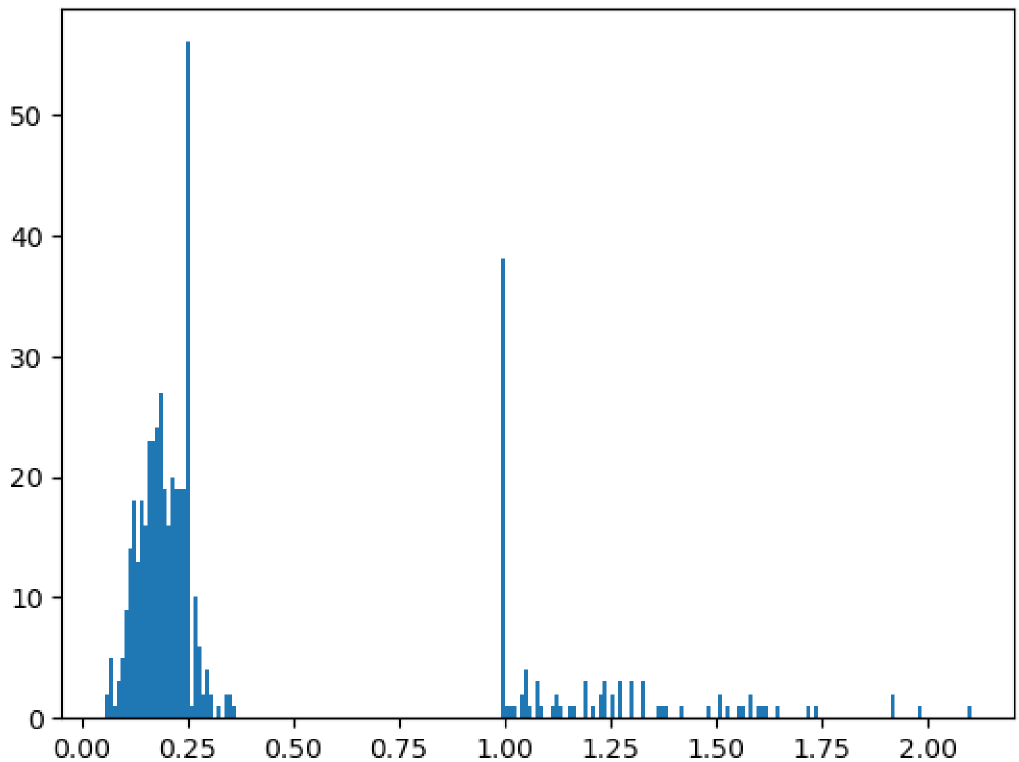
[6]:
```
u = cp.Variable(other, 'u')
dtumor = cp.Variable(tumor, 'dtumor')
dother = cp.Variable(other, 'dother')
b = cp.Variable(n, 'b')
f = cp.maximum(dother - Dother, 0)
obj = cp.sum(f)
cons = [Aother @ b == dother, Atumor @ b == dtumor, dtumor >= Dtarget, b >= 0,␣
 ↪Bmax >= b, u >= 0, u >= dother - Dother]
problem = cp.Problem(cp.Minimize(obj), cons)
problem.solve(verbose = True, solver = cp.ECOS)
```

```
===============================================================================
                                     CVXPY
                                     v1.4.2
===============================================================================
(CVXPY) Mar 19 06:40:33 PM: Your problem has 1200 variables, 7 constraints, and
0 parameters.
(CVXPY) Mar 19 06:40:33 PM: It is compliant with the following grammars: DCP,
DQCP
(CVXPY) Mar 19 06:40:33 PM: (If you need to solve this problem multiple times,
but with different data, consider using parameters.)
(CVXPY) Mar 19 06:40:33 PM: CVXPY will first compile your problem; then, it will
invoke a numerical solver to obtain a solution.
(CVXPY) Mar 19 06:40:33 PM: Your problem is compiled with the CPP
canonicalization backend.
-------------------------------------------------------------------------------
                                  Compilation
-------------------------------------------------------------------------------
(CVXPY) Mar 19 06:40:33 PM: Compiling problem (target solver=ECOS).
(CVXPY) Mar 19 06:40:33 PM: Reduction chain: Dcp2Cone -> CvxAttr2Constr ->
ConeMatrixStuffing -> ECOS
(CVXPY) Mar 19 06:40:33 PM: Applying reduction Dcp2Cone
(CVXPY) Mar 19 06:40:33 PM: Applying reduction CvxAttr2Constr
(CVXPY) Mar 19 06:40:33 PM: Applying reduction ConeMatrixStuffing
(CVXPY) Mar 19 06:40:33 PM: Applying reduction ECOS
(CVXPY) Mar 19 06:40:33 PM: Finished problem compilation (took 2.240e-02
seconds).
-------------------------------------------------------------------------------
                                Numerical solver
-------------------------------------------------------------------------------
(CVXPY) Mar 19 06:40:33 PM: Invoking solver ECOS  to obtain a solution.
-------------------------------------------------------------------------------
                                    Summary
-------------------------------------------------------------------------------
(CVXPY) Mar 19 06:40:33 PM: Problem status: optimal
(CVXPY) Mar 19 06:40:33 PM: Optimal value: 1.388e+00
(CVXPY) Mar 19 06:40:33 PM: Compilation took 2.240e-02 seconds
(CVXPY) Mar 19 06:40:33 PM: Solver (including time spent in interface) took
```

```
1.834e-01 seconds
```

[6]:
```
1.3882005424141164
```

[7]:
```python
#d = np.concatenate(np.array(dother.value), np.array(dtumor.value))
d = [0] * (tumor+other)
for i in range(tumor):
    d[i] = dtumor.value[i]
for i in range(other):
    d[i+tumor] = dother.value[i]
plt.hist(d, bins=225)
plt.xticks(np.arange(0, 2.25, 0.25))
plt.show()
```



So both of these optimization problems give the same solution.